REPORTAGE ESCLUSIVO
OLIMPIADI DELL'INFORMATICA

PASSO PASSO: UN APPLICATIVO PER IDENTIFICARE CHI TI TELEFONA

VERSIONE PLUS RIVISTA+LIBRO+CD €14,90
VERSIONE STANDARD RIVISTA+CD €6,90

Periodicità mensile • NOVEMBRE 2003 • ANNO VII, N.10 (74)
Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL

ioPROGRAMMO

IN ALLEGATO

DAI VOCE ALLE APPLICAZIONI

Visual Basic: come sviluppare un software che parla e obbedisce ai comandi vocali

- Sintesi della voce
- Nuove tecniche di riconoscimento
- Il tutorial completo

Java multimediale

Realizzare un player audio in stile Winamp

TRATTAMENTO DEL TESTO

Le Regular Expression diventano facili con .NET

IMPARARE GIOCANDO...

Applicazioni distribuite: la briscola in rete

SISTEMA
- Trasformare un documento XML in codice C#

NETWORKING
- Accesso multicanale ai dati con Flash MX e Web Service
- VB: l'accesso alle risorse di rete

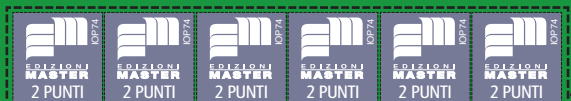
PALMARI
- Pocket PC: operazioni remote sul server database

ELETTRONICA
- La rotazione del polso in un braccio meccanico
- La programmazione del Sony AIBO

ADVANCED
- Delphi 7: attachment in SOAP

CORSI
- Visual Basic, Java, C++, VB.NET, C#, MATLAB

ISSN 1128-594X
9 771128 594641 30074
ioProgrammo (plus) Anno VII - N° 10 (74) • € 14,90
EDIZIONI MASTER
www.edmaster.it

SCOPRI LA NUOVA SEZIONE CHE METTE ALLA PROVA IL TUO INTUITO DI PROGRAMMATORE

Anno VII - n. 10 (74) Novembre 2003

▼ Uscire dalla crisi

Scrivo queste note dallo SMAU, ultimo giorno di fiera. Chi di voi ha avuto l'opportunità di passare e dare uno sguardo agli stand presenti, si sarà certamente accorto della estrema riduzione, rispetto agli anni passati, del numero di aziende presenti. La crisi del settore informatico e quella più generale dell'economia italiana, possono spiegare solo in parte la riduzione degli investimenti che ha ridotto SMAU ad una pallida ombra dell'evento che fu. La radicale trasformazione dell'informatica, partita come settore autonomo e divenuta lievito onnipresente e determinante per l'intero mondo tecnologico, non è stata probabilmente colta da molti dei principali operatori. A fronte di un netto calo dei visitatori (alla fine credo arriveremo ad un -15%), c'è da registrare un aumento di presenze fra gli addetti al settore, testimonianza della crescita dell'interesse nell'informatica come business. Meno lucette e più sostanza: potrebbe essere lo slogan di questa manifestazione che, nel bene e nel male, resta uno specchio fedele dello stato dell'informatica in Italia. Qui a ioProgrammo continueremo a seguire la nostra strada, alla ricerca del lato piacevole della tecnologia, sempre attentissimi ai vostri suggerimenti. Fatevi sentire!

P.S.

Non possiamo non ringraziarvi per l'attenzione che state dedicando al nostro ultimo nato: il sito di ioProgrammo continua a crescere grazie al vostro interesse e nuove iniziative stanno per essere varate. Stay tuned...

raffaele@edmaster.it

Raffaele del Monaco

CD WEB
nome_file.zip
cdrom.ioprogrammo.it

All'inizio di ogni articolo, troverete un nuovo simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it. Per scaricare software e codice da Internet, ogni mese indicheremo una password differente. Per il numero che avete fra le mani la combinazione è:

Username: **neo** *Password*: **matrix**

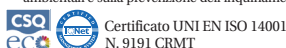
ioProgrammo

Anno VII - N.ro 10 (74) - Novembre 2003 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale Massimo Sesti
Direttore Responsabile Romina Sesti
Responsabile Editoriale Gianmarco Bruni
Responsabile Marketing Antonio Meduri
Editor Gianfranco Forlino
Coordinamento redazionale Raffaele del Monaco
Redazione Antonio Pasqua, Thomas Zaffino
Collaboratori M. Autiero, L. Buono, M. Canducci, A. Cangiano, M. Del Gobbo, F. Grimaldi, F. Lippo, R. Lo Storto, D. Magoga, A. Marroccelli, P. Martemucci, S. Monni, G. Naccarato, C. Pelliccia, P. Perrotta, L. Salerno, F. Sara, L. Spuntoni, E. Tavolaro, S. Tura F. Vaccaro, D. Vesicchio, V. Vessia,
Segreteria di Redazione Veronica Longo

Realizzazione grafica Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale SET S.r.l.
Coordinamento Tecnico Piero Mannelli
Realizzazione CD-Rom Paolo Iacona

Pubblicità Master Advertising s.r.l.
Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Rete Vendita Serenella Scarpa, Cornelio Morari, Roberto Piano, John F. Alborante
Segreteria Ufficio Vendite Daisy Zonato

Editore Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121206
Sede di Rende: C.da Lecco, zona industriale - 87030 Rende (CS)
Amministratore Unico: Massimo Sesti

Abbonamento e arretrati
ITALIA: Abbonamento Annuale: ioProgrammo Basic(11 numeri): € 52,90 sconto 30% sul prezzo di copertina € 75,90.
ioProgrammo Plus (11 numeri + 6 libri): € 86,90 sconto 30% sul prezzo di copertina € 123,90.
ESTERO: Abbonamento Annuale: ioProgrammo Basic (11 numeri): € 151,80. ioProgrammo Plus (11 numeri + 6 libri): € 257,00
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via Cesare Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- Bonifico bancario intestato a Edizioni Master S.r.l. c/o Banca Credem S.p.a. c/c 5000 ABI 03032 CAB 80880 (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a: Edizioni Master Servizio Clienti - Via Cesari Correnti, 1 - 20123 Milano
Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel.02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Rotoeffe Via Variante di Cancelleria, 2/6 - Ariccia (Roma)
Stampa CD-Rom: Deluxe Italy S.r.l. - via Rossini, 4 - Tribiano (MI)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Ottobre 2003

Edizioni Master edita:
Idea Web, Go!OnLine Internet Magazine, Win Magazine, PC Fun extreme, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all'Home Cinema, <tag/>, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, La mia videoteca, Le Collection.

NOVITÀ PER LE APPLICAZIONI MULTIMEDIALI

Norpath ha rilasciato la versione 2.1 del suo *Norpath's Norpath Elements Studio,* un software di authoring che permette di creare e distribuire applicazioni multimediali interattive.
Tra le caratteristiche:

un **editor di temi** disponibile a run-time per cambiare al volo il look&feel delle applicazioni;
un **sistema di monitoraggio** per le azioni compiute dagli utenti;
un nuovo **sistema di database** indirizzato alla creazione di quiz e test;
la possibilità di inviare **e-mail dalle applicazioni** che si vanno a realizzare.

www.norpath.com

NUOVI ORIZZONTI PER JAVA E XML

NetKernel, un innovativo sistema operativo virtuale Open Source, offre una nuova piattaforma per la manipolazione di XML. Basato su Java, consente un'estrema flessibilità nel trattamento e nella visualizzazione di dati XML: tutto il processo di sviluppo e deploy di processi XML sono ora possibili con grande razionalità e semplicità.
È incluso il supporto al *Simple Tree Manipulation Language* (STM) e a *XPath Document API* (XDA). Inoltre, grazie alla sua modularità, può facilmente essere esteso e anche personalizzato.

www.1060research.com

News

METANOLOGY ENTRA IN ECLIPSE

Un nuovo membro si aggiunge alla prestigiosa e ormai lunga lista di aziende che supportano il consorzio Eclipse.
In contemporanea con l'ingresso in Eclipse, Metanology ha annunciato la nuova release di MDE (Model-driven Development Environment) uno strumento di viluppo model-driven, basato su Eclipse, conforme alle specifiche fornite dall'Object Management Group's Model Driven Architecture (MDA).

www.metanology.com

ESPIAL INTEGRA FLASH

Macromedia ed Espial hanno raggiunto un accordo per integrare la tecnologia Flash nel browser Espial Escape, appositamente realizzato per dispositivi embedded.

espial escape™

Espial Escape è stato scelto da produttori del calibro di Intel, Motorola, NEC, M@gicMedia, Siemens e DaimlerChrysler, e l'accordo siglato per l'integrazione di Flash apre nuove strade al gioiello Macromedia ed ai suoi sviluppatori.

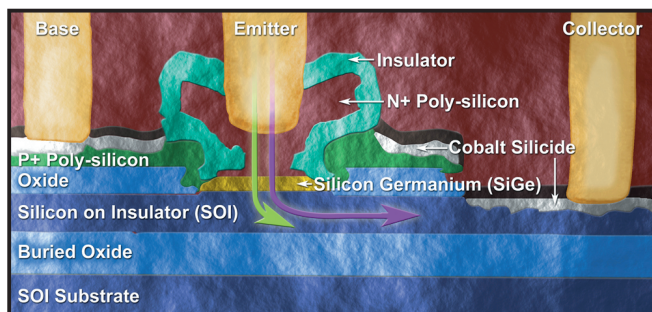
www.espial.com

WINDOWS XP

Pronta la versione beta nativa a 64 bit del sistema operativo Windows XP, progettata per supportare i sistemi a 64 bit, incluse le piattaforme AMD basate su tecnologia AMD64
Il sistema operativo aggiornato a 64 bit, chiamato Windows XP 64-Bit Edition for 64-Bit Extended Systems è progettato per supportare i sistemi a 64 bit, incluse le piattaforme AMD basate su tecnologia AMD64. Il sistema operativo funzionerà su desktop con processore AMD Athlon 64 e workstation con processore AMD Opteron.
Un vantaggio fondamentale del nuovo sistema operativo è la tecnologia Microsoft Windows on Windows 64 (WOW64), che consentirà ai clienti attualmente in possesso di applicazioni a 32 bit compatibili con Windows XP, di utilizzare un sistema operativo a 64 bit. L'architettura WOW64 sfrutta l'architettura AMD64 rendendo compatibili le applicazioni a 32 bit e permettendone l'esecuzione con un buon livello di prestazioni.
"Abbiamo sentito dire dai nostri clienti che, fino ad ora, il maggiore ostacolo nell'investimento su tecnologie a 64 bit è stata l'impossibilità di far funzionare in maniera efficiente le applicazioni a 32 bit su sistemi a 64 bit", ha affermato Fabrizio Albergati, direttore Gruppo Windows Client e Mobility di Microsoft Italia. "Combinando Windows XP ed i nuovi processori

SMALL COMPUTING NEL FUTURO BIG BLUE

I laboratori dell'IBM stanno lavorando su una nuova tecnologia che potrebbe portare grandi benefici al mercato wireless. Fino ad ora i chip per la trasmissione dei dati erano distinti e non integrabili con quelli riservati alla elaborazione. Le due tipologie si avvalgono infatti di leghe di silicio diverse. Con la nuova tecnologia che IBM sta mettendo a punto, questa dicotomia si potrebbe sanare e potremo presto vedere integrati su un singolo chip tutte le capacità di calcolo e trasmissione necessarie al funzionamento di un dispositivo wireless. Ai vantaggi in termini di dimensioni, si andranno ad aggiungere anche vantaggi in termini di autonomia: i nuovi chip consumeranno fino a cinque volte meno degli attuali.

www.ibm.com

La nuova tecnologia per la costruzione di chip.

64 BIT IN BETA

AMD64, è possibile ottenere tutta la potenza e la memoria necessaria, pur continuando a utilizzare le stesse applicazioni".

Windows XP 64-Bit Edition per Extended Systems a 64-Bit è stato progettato per essere utilizzato su computer ad elevate prestazioni. In altre parole, gli utenti potranno superare il limite della memoria fisica, pari a 4 gigabyte, di un computer a 32 bit.

La potenza della nuova piattaforma Windows renderà possibili nuove esperienze sul proprio PC, soprattutto nell'area dell'advanced gaming, della creazione di contenuti digitali e del video editing. Gli appassionati di videogiochi entreranno in una nuova dimensione di realismo, mentre gli esperti di digital media saranno in grado di creare contenuti con una qualità normalmente raggiunta soltanto dai professionisti.

La principale differenza tra l'elaborazione basata su Windows XP a 32 piuttosto che a 64 bit risiede nella capacità della versione a 64 bit di utilizzare una quantità notevolmente superiore di memoria di sistema. La versione a 64 bit di Windows XP supporterà inizialmente fino a 16 GB (gigabyte) di RAM e un massimo di 8 TB (terabyte) di memoria virtuale.

www.microsoft.com

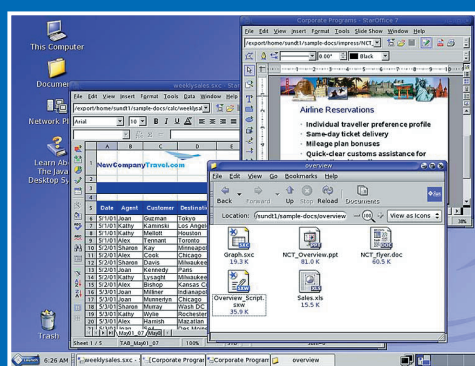
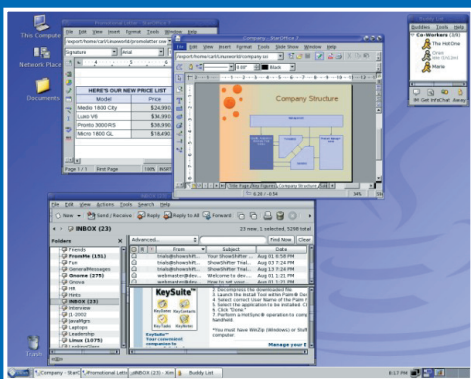
UN FUTURO NUOVO PER LE CABINE TELEFONICHE

La British Relecom ha iniziato ad installare degli *Access Point Wi-Fi* nelle cabine telefoniche pubbliche della sua rete. Oltre 108.000 sono le cabine di proprietà BT, ma solo una parte verranno rese Wi-Fi enabled, seguendo una strategia che privilegia le zone a più intenso affollamento di uffici, i cui dipendenti possano effettivamente approfittare di questa opportunità.

Le nuove installazioni saranno a prova di vandali, possono resistere ai rigidi inverni londinesi e, grazie alle dimensioni ridotte, potranno montate in modo invisibile nelle vecchie cabine rosse che ancora caratterizzano parte dell'Inghilterra.

LA SFIDA DI SUN

Il dominio del mercato dei sistemi desktop è saldamente nelle mani di Microsoft, eppure qualcosa potrebbe cambiare.

Almeno stando alle dichiarazioni di Sun che ha finalmente svelato cosa si celava dietro il progetto Mad Hatter: una completa suite di strumenti per il desktop computing, sia per piattaforma x86 che SPARC su sistemi operativi Solaris o Linux, e che annovera StarOffice 7, Mozilla, la piattaforma di sviluppo J2SE, applicazioni per la posta elettronica e per il messaging, insomma tutto,

I moduli sono la novità più ghiotta.

o quasi, si configuri come necessario per un utilizzo tipico del PC. Come partner di questa iniziativa, nonché fornitori di importanti tasselli software, troviamo nomi del calibro di Computer Associates, Adobe e RealNetworks, tutti fiduciosi in nella possibilità di fare breccia nell'impero Microsoft. Java Desktop System, questo il nome della suite, avrà come asso nella manica il prezzo di appena 100 dollari e sarà disponibile da dicembre 2003.

www.macromedia.it

APPLICAZIONI .NET SU LINUX? ORA È REALTÀ!

Novell è certa che entro la fine dell'anno sarà pronta a lanciare sul mercato la prima versione commerciale della tecnologia che consente di far girare applicazioni MS.NET su Linux. MONO, di cui già si è sentito parlare molto, sarà rilasciato in via definitiva entro la fine dell'anno. Questo il comunicato rilasciato da Novell.

Il progetto Mono, frutto di un intenso lavoro di sviluppo, si compone di un compilatore per il linguaggio C# e VB.NET con cui è possibile sviluppare applicazioni compatibili con la piattaforma di Microsoft; un compilatore just-in-time Common Language Runtime, che consente a Linux di far girare applicazioni scritte sotto un qualsiasi altro sistema operativo che supporti MS.NET.

www.novell.com

L'ITALIA POTRÀ VISIONARE IL CODICE SORGENTE DI WINDOWS

Il Ministro dell'Innovazione e le Tecnologie, Lucio Stanca, ha dichiarato di aver siglato con Microsoft il protocollo d'intesa *"Government Security Program"*, che consente al CNIPA *(Centro Nazionale per l'Informatica nella Pubblica Amministrazione)*, l'accesso di codice.
Secondo quanto dichiarato dal Ministro, l'accordo con il big di Redmond è strategico e importante per la sicurezza e l'interoperabilità dei sistemi informatici, una tappa miliare per lo sviluppo delle nuove applicazioni tecnologiche di cui beneficerà anche il nostro Paese".

MICROSOFT LANCIA IL SITO WEB OFFICE-INFOSITE

Dopo l'ok alla produzione di Office 2003, Microsoft ha messo online un portale dedicato alla nuova versione. Si tratta di un portale contenente news, help-online, training, aggiornamenti, add-on e plug-in per la nuova release di Office 2003.
Chris Linnet, Group Manager del sito, ha dichiarato che una buona parte delle risorse saranno direttamente utilizzabili e scaricabili delle applicazioni Office, senza dover impiegare il browser, grazie ai sistemi di transazione garantiti dal supporto XML.

http://office.microsoft.com/

DOPO 20 ANNI, IL COFONDATORE DI SUN MICROSYSTEMS ABBANDONA

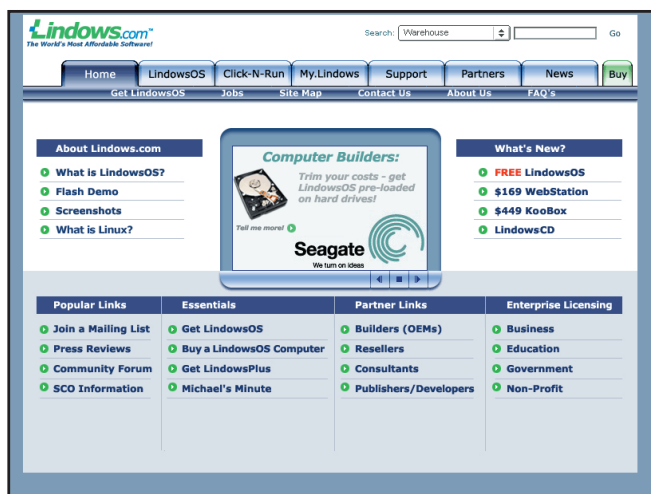
Bill Joy, attuale Chief Scientist della società, lascia la Sun. A lui si devono molte delle rivoluzionare idee: Solaris, i microprocessori Sparc e Java, la versione Berkeley del sistema operativo Unix (BSD), questi alcuni dei progetti ai quali Bill Joy ha dedicato ben 20 anni di lavoro. Un vero genio dell'informatica che ha così commentato la sua decisione: *"Dopo 21 anni ho apprezzato l'opportunità di creare innovazione fornitami da Sun, ma ho deciso che è tempo di muoversi su differenti sfide"*.

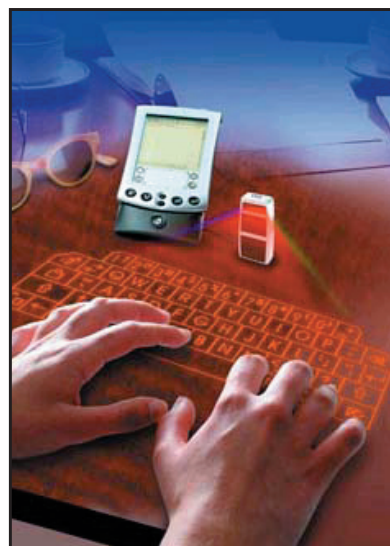
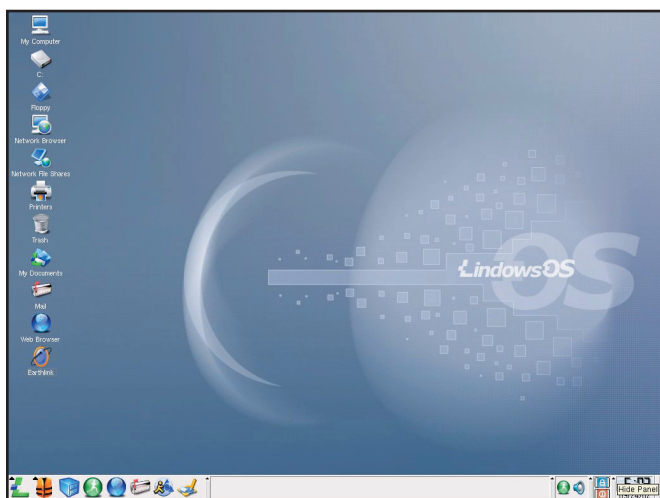
SUSE LINUX 9.0 PRENDE LA LAUREA

La nuova versione del sistema operativo di casa SUSE riceve ufficialmente la certificazione LSB già prima della messa in commercio del prodotto, confermando la piena compatibilità del nuovo sistema operativo con gli standard universali dettati dal mondo Linux. LSB, il gruppo di lavoro attivo all'interno del Free Standard Groups e che si occupa della verifica della compatibilità agli standard dei prodotti rispetto alla politica di Open Source Linux e dei tools su base Gnu, ha sviluppato una stretta collaborazione tra la comunità di sviluppatori, le aziende Linux, gli ISV e i maggiori vendor presenti sul mercato, garantendo ai clienti, attraverso la certificazione di un prodotto, il pieno rispetto degli standard di compatibilità e di sicurezza sui quali si fonda Linux.

www.opengroup.org/lsb/cert/

LINDOWS 4.0 ARRIVA IN ITALIA

Il sistema operativo, un ibrido tra Windows e Linux, arriverà in Italia in modo ufficiale con la sua più recente versione: LindowsOS 4.0
Le società Lindows.com e Questar commercializzeranno e promuoveranno la vendita sul mercato italiano della versione inglese di LindowsOS 4.0, il siste-

USA LA TASTIERA CHE NON C'È!

La società israeliana VKB presenta una sorta di tastiera "fantasma": una rappresentazione olografica di una tastiera. Via quindi le vecchie, pesanti e ingombranti tastiere!
Grazie alla VKB, per mezzo di un particolare laser che proietta l'immagine di tutti i tasti di una comune tastiera, è possibile digitare direttamente su una qualsiasi superficie piana. Il sistema è ovviamente in grado di identificare il tasto virtuale premuto dall'utente. E' possibile usare la tastiera con un PC, un palmare o un telefonino cellulare. Fantastico!

www.vkb.co.il

Il desktop della nuova versione di LindowsOS.

ma operativo Linux-based che consente di far "girare" gran parte delle applicazioni Windows avendo a disposizione, comunque, un sistema Linux.
Tra i molti strumenti proposti all'interno dell'ultima versione di Lindows OS, spicca Click-N-Run, il tool che consente di scaricare, installare e lanciare in modo automatico un'applicazione qualsiasi, tra le oltre 1.800 messe a disposizione da Click-N-Run Warehouse.

www.lindows.com

DA HP UNA FOTOCAMERA CHE TRADUCE LE SCRITTE

Sebbene sia ancora solo un prototipo, questo aggeggio potrebbe rivoluzionare la vita di molte persone. Una normale macchina fotocamera digitale ma con potenzialità enormi: tradurre le scritte dalle immagini che ritrae. Cartelli stradali, menu al ristorante, ricevute. La telecamera è in grado di collegarsi ad un sito Internet, fornendo allo stesso l'immagine contenente il testo da tradurre e "aspettando" una risposta contenente la traduzione. Del prodotto, ancora in via sperimentale, non è stata resa pubblica la data di commercializzazione.

www.hp.com

SONY ERICSSON Z600

65000 colori, fotocamera incorporata, bluetooth, 32 suonerie polifoniche, effetto force-feedback (in stile controller PS2) per i videogame. Sony Ericsson, in occasione di un evento dedicato alla stampa internazionale, ha presentato una gamma di nuovi telefonini cellulari; tra questi lo Z600, il più sofisticato dei tre telefoni cellulari presentati. La vera novità è data dal controller force-feedback, ovvero una versione ridotta del classico controller per PlayStation, chiamata Gameboard EGB-10.
Si aggancia alla base dello Z600, fornendo all'utilizzatore quattro tasti (in stile PS2) e un pulsante per il pollice.

www.sonyericsson.com

BORLAND UTILIZZERÀ IL FRAMEWORK WXWINDOWS

Per la sua nuova linea di prodotti C++ BuilderX Borland adotterà il framework wxWindows, capace di fornire un nutrito set di API per scrivere applicazioni GUI su diverse piattaforme.
Così facendo, Borland conferma il suo interesse al mondo Open Source.
Inoltre, Borland ha annunciato la release di Together per piattaforma .NET. L'annuncio è avvenuto in occasione della conferenza VSLive di Orlando. Todd Olson, chief scientist di Together in Borland, ha dichiarato che il nuovo prodotto è sviluppato completamente in linguaggio C#, questo per garantire la massima compatibilità di Together con la nuova piattaforma .NET.
Together è un ambiente completamente integrato per la creazione di modelli, destinato ai team che producono software e che necessitano di accelerare il ciclo di sviluppo applicativo.

www.borland.com

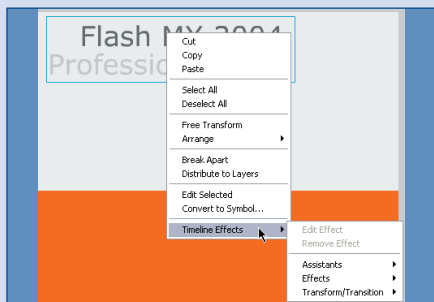
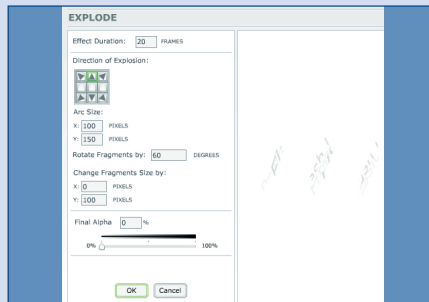
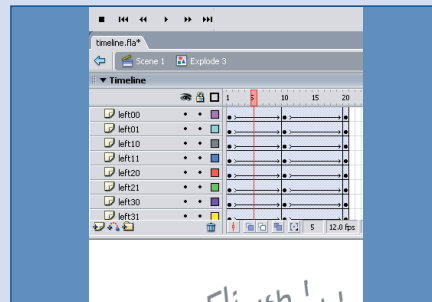
Flash MX 2004 Professional

La versione completa del più celebre applicativo per la creazione di progetti multimediali, ancora più semplice e potente.

Dopo mesi di voci non confermate e dichiarazioni ufficiali sulle nuove caratteristiche tecniche, il 10 Settembre 2004 la Macromedia ha messo finalmente a disposizione i nuovi prodotti MX nella versione inglese. Come di consueto, le sorprese non sono mancate: un player più veloce, nuovi strumenti di sviluppo e ben due nuove versioni, una dedicata ai designer (Flash MX 2004) ed una pensata per i developer (Flash MX 2004 Professional). Questa scelta di realizzare due pacchetti distinti testimonia la divisione di ruoli che la Macromedia ha individuato nell'utilizzo della tecnologia alla base di Flash, e che possiamo definire l'evoluzione naturale di una strategia di cui si potevano cogliere i primi segnali a partire dalla versione MX. La versione Professional possiede tutte le caratteristiche della versione normale con in più dei componenti aggiuntivi creati per agevolare lo scambio di dati e numerose altre opzioni, molte delle quali pensate per dialogare con programmi di terze parti.

LA NUOVA INTERFACCIA

Appena lanciamo l'applicazione appare la Start Page. Questa finestra, oltre a fornire una guida sulle novità più rilevanti, mostra a sinistra un elenco dei nuovi templates e al centro, sotto la voce *create new*, una nuova tipologia di documenti, nonché una serie di opzioni che hanno il compito di agevolare l'utente nell'uso delle procedure più utilizzate. Una volta aperto un nuovo file, la Start Page lascia il posto al nuovo ambiente di lavoro (Fig.1), caratterizzato da una nuova veste grafica: nuove icone, nuovi colori e una nuova disposizione dei contenuti. L'impressione generale resta, comunque, quella di un layout abbastanza fedele a quello precedente, nonostante i gradevoli cambiamenti stilistici. Inoltre - a proposito di grafica - per quanto riguarda gli strumenti di disegno, risulta quasi invariata la Tools bar; l'unica novità rilevante è lo strumento *Polystar*, presente nel menu a comparsa attivabile dallo strumento *Rectangle*. Discorso diverso, invece, per la disposizione ed i contenuti dei pannelli, dove troviamo non poche novità. Esplorando infatti la voce *Window*, notiamo che tutti i pannelli sono stati riorganizzati in tre sottogruppi:

UTILIZZARE EFFETTI GRAFICI PREIMPOSTATI CON I TIMELINE EFFECTS

1 **A seconda dell'effetto scelto, possono essere applicate suddivisioni in più livelli, duplicazioni, interpolazioni o altro, il tutto senza dover nemmeno sfiorare la timeline. Facciamo un esempio, con un simbolo grafico costituito da una semplice scritta. Una volta posizionato il simbolo sul primo fotogramma della linea temporale, selezioniamolo con il tasto destro del mouse e attiviamo la voce *Effects*. Dall'elenco di effetti disponibili, scegliamo *Expolde*.**

2 **Ci appare una nuova finestra di dialogo che ci chiede di regolare le impostazioni dell'esplosione. E' possibile regolare una serie di paramentri, come ad esempio la durata, la direzione, la rotazione degli elementi, e così via. A destra c'è una piccola anteprima dell'effetto, aggiornabile cliccando sull pulsante *Update Preview*, per capire come agiranno le nostre regolazioni. Lasciamo le opzioni di default e premiamo *ok*.**

3 **Una volta premuto *ok*, se osserviamo la timeline, notiamo che sono stati occupati 20 fotogrammi della linea temporale. Inoltre, nella libreria il nostro simbolo grafico originario è stato rinominato come *Explode3*. Se clicchiamo sul simbolo con il tasto destro del mouse e attiviamo la voce *edit*, possiamo verificare cosa è successo nella timeline del simbolo: Flash ha creato automaticamente più di 30 livelli e altrettante interpolazioni.**

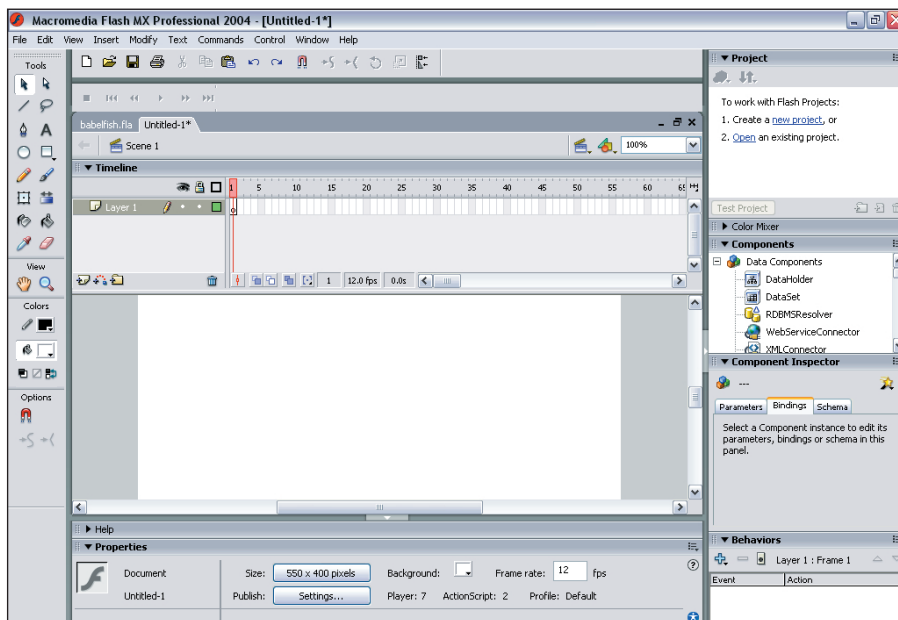
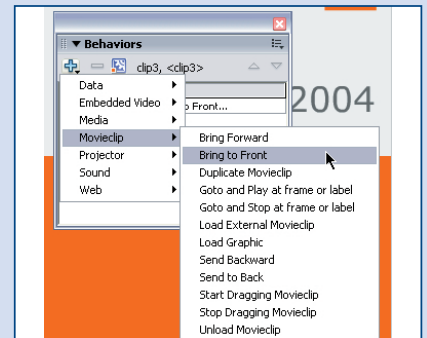
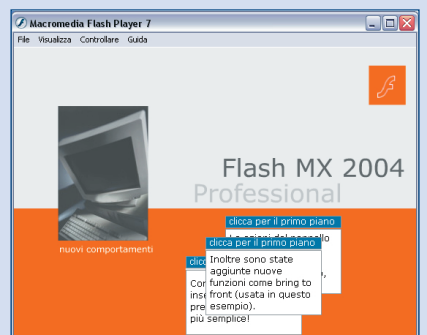



Fig. 1: *La nuova interfaccia semplifica tutte le operazioni cui eravamo abituati.*

il gruppo Design panel, che comprende tutti i pannelli relativi agli strumenti di disegno.
il gruppo Development panel, con i pannelli relativi alla programmazione.
il gruppo Others panel, che raggruppa pannelli generali non catalogabili nelle due precedenti categorie.

Bisogna specificare che, nella maggior parte dei casi, si tratta di una semplice riorganizzazione di pannelli già esistenti, anche se non mancano nuovi arrivi, come ad esempio il pannello *History*, attraverso il quale è possibile ripercorrere i vari "passi" compiuti, cioé le varie fasi di un lavoro. Ci sono, tuttavia, anche pannelli già esistenti, ma completamente ripensati, come il nuovo pannello *Help*. Una piccola rivoluzione riguarda il pannello Actions (ora collocato nel gruppo dei *Development panel*), che presenta parecchie novità, la più eclatante di tutte consiste nella totale abolizione della modalità di scrittura *normal mode*. Come tutti gli utenti delle versioni precedenti sanno, quando si apriva il pannello *Actions*, la *normal mode* (nelle versioni italiane *modalità normale*), attraverso una finestra di dialogo, aiutava a scrivere il codice mediante vari campi che, il più delle volte, prevedevano valori preimpostati tra cui scegliere. Un modo comodo e funzionale, di elaborare script, anche abbastanza complessi, senza mettere direttamente mano al codice. Adesso, invece, dei vari automatismi presenti nella precedente versione, è sopravvissuta solo la possibilità di prelevare e trascinare dall'elenco a sinistra (l'*actions toolbox*) i vari metodi e proprietà, per poi posizionarli nell'*action panel*. In altre parole, si può scrivere il codice solo nella modalità che conoscevamo come *expert mode*. Un'altra significativa novità riguarda la presenza di una nuova sezione, posta a sinistra, sotto il consueto elenco di istruzioni: stiamo parlando del menu *Script navigator*, una novità che consente di esplorare i codici inseriti all'interno di diverse scene, fotogrammi o clip, senza ricorrere alla timeline. Questa scelta evidenzia la volontà della Macromedia di relegare la timeline ad un settore prettamente ed esclusivamente grafico, scindendola dal gruppo di strumenti destinati allo sviluppo. In ogni caso, non sono state dimenticate le esigenze di quegli utenti che gradiscono una guida nella stesura del codice: di fatto, il nuovo pannello *Behaviors*, che si può aprire con il percorso *Window>Development panel>Behaviors*, consente di inserire un discreto pacchetto di codici ActionScript, in modo del tutto automatico. Il pannello è contestuale, per cui, se si seleziona ad esempio un fotogramma con il pannello aperto, nell'elenco delle istruzioni disponibili compariranno solo quelle valide per i fotogrammi e così via. Indubbiamente questo approccio è meno formativo rispetto alla vecchia modalità *normal*, che consentiva di inserire un codice più complesso seguendo vi-

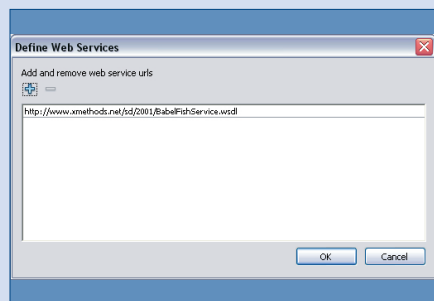
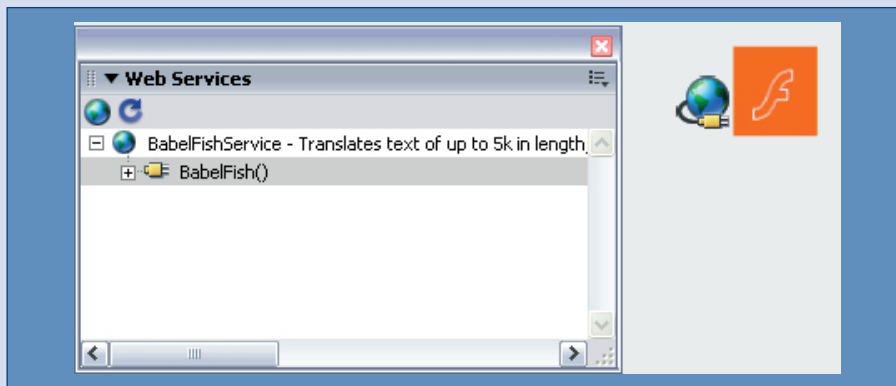
IL PANNELLO BEHAVIOR

1 **Il pannello *Behavior*, oltre a sostituire la modalità normale, prevede alcune nuove funzioni estremamente utili. Adesso infatti è possibile ordinare i livelli di un clip, anche senza ricorrere al metodo *swapDepth()*. Prima di tutto, creiamo tre clip, che denomineremo rispettivamente *clip1*, *clip2* e *clip3*. I tre clip devono essere posizionati sullo stage, nello stesso fotogramma del secondo livello, in modo da essere parzialmente sovrapposti. Selezioniamo il primo clip, in modo da creare un riferimento che permetta al pannello Behaviors di mostrare all'utente l'elenco di istruzioni valide solo per i MovieClip. Seguiamo quindi il percorso *Windows>Development panel>Behaviors* e clicchiamo il pulsante *Add Behavior*, contraddistinto dal segno "+". Dal menu a comparsa, selezioniamo *Movieclip*: così facendo appare tutto l'elenco di azioni valide per i clip previste nel pannello in questione.**

2 **Selezioniamo l'azione *Bring to front*: così facendo, al primo clip viene aggiunto un evento *On Release* che, se innescato, lo posiziona automaticamente in primo piano, a prescindere dalla sua posizione iniziale. Per fare in modo che ogni clip venga posto in primo piano ad ogni evento *On Release*, bisogna ripetere lo stesso procedimento anche per gli altri due.**

sivamente le varie fasi della sua composizione. Oggi, un nuovo utente che dovesse iniziare l'apprendimento di Flash, usando esclusivamente il pannello *Behavior*, rischierebbe di attuare delle soluzioni senza comprenderne a fondo il meccanismo. Un discorso paragonabile ai problemi che incontra chi impara ad usare *DreamWeaver*, senza aver mai scritto una riga di HTML o JavaScript.

ACTIONSCRIPT 2.0

Continuiamo con le grosse novità. A partire dalla versione 2004, nasce ActionScript 2.0 che, oltre a nuovi oggetti, metodi e proprietà, introduce alcune convenzioni, obbligatorie in alcuni casi e solo consigliate in altri. Prima di tutto, i nomi delle variabili o delle funzioni diventano tutti case-sensitive (la variabile *miavar* è diversa da *MiaVar*). Inoltre, i dati sono diventati rigidamente tipizzati: ad esempio, quando si dichiara una variabile, bisogna specificare il tipo di dato, in modo simile a quanto viene fatto in Java con una sintassi del tipo: *var MiaVariabile:String="esempio"* - e non semplicemente *miaVariabile="esempio"* come accadeva in ActionScript 1.0. Adesso è anche possibile scrivere vere e proprie classi, o aggiungere sottoclassi a classi già esistenti, creando dei file esterni con estensione *.as*. In pratica, la classe si scrive con il nuovo pannello *Script (File>New>ActionScript file)*, che agevola la realizzazione di script posti all'esterno del filmato *swf* - mentre chi usa la versione MX 2004 "semplice" deve usare un normale programma di videoscrittura. Anche se gli sviluppatori abituati ad usare ActionScript 1.0 potrebbero avere qualche iniziale diiffidenza, non c'è motivo di preoccuparsi troppo (almeno con la versione attuale): nei fatti, il nuovo ActionScript convive perfettamente con il vecchio che, per le funzionalità già esistenti, non ha l'obbligo di rispettare le nuove convenzioni. Anche l'uso del versatile prototype, che consentiva di aggiungere metodi personalizzati ai propri oggetti e MovieClip, adesso convive con i nuovi procedimenti. Lo stesso file *swf*, del resto, è compilato ancora in ActionScript 1.0. Il giudizio su Flash MX 2004 Professional è sostenzialmente positivo. La rinnovata potenza, l'utilità dei componenti aggiunti (è possibile farsi un'idea dei nuovi datacomponents attraverso l'esempio *Babelfish.swf* disponibile sul CD allegato alla rivista) e la qualità

I DATACOMPONENTS

Una delle novità di Flash MX 2004 Professional, è l'utilizzo del componente WebServiceConnector per dialogare con i web service. Alcuni degli utenti che per primi si sono cimentati con i nuovi componenti di Macromedia, nonostante quest'ultima abbia messo a disposizione sul suo sito accurate note tecniche, hanno riscontrato delle difficoltà.
Facciamo quindi un po' di chiarezza: il componente in questione funziona perfettamente, ma è contraddistinto da alcune restrizioni che, per motivi di sicurezza, impediscono la fruizione completamente automatica di un qualsiasi web service. Facciamo un esempio concreto, creando una semplice interfaccia Flash che funga da traduttore dall'italiano all'inglese, grazie al web service del celebre BabelFish di AltaVista. Innanzitutto, per provare il file swf (sia quello che realizzeremo assieme, sia quello completo, disponibile negli esempi) è necessario essere connessi ad Internet.

1 **Apriamo il pannello relativo alla gestione dei web services: *Window>Development Panels>Web services.***
Premiamo il pulsante *Define Web Services* (che ha la forma di un piccolo pianeta), in modo da aprire la relativa finestra di dialogo.
Nella finestra, dobbiamo premere il pulsante Add Web Services e, successivamente, inserire nel campo che compare nello spazio sottostante l'indirizzo del nostro web service: *http://www.xmethods.net/sd/2001/BabelFishService.wsdl*

2 **Al termine di questa operazione, nel pannello *Web Services* appare la voce *BabelFishService*. Selezioniamo con il mouse il piccolo segno + accanto alla voce, e clicchiamo con il tasto destro sul metodo *Bablefish()*, che compare in basso. Ci apparirà un menu con alcune voci, tra cui quella che dobbimo scegliere: *Add method Call*. In questo modo si creerà automaticamente sullo stage un'istanza del componente *WebServiceConnector*. Al componente, diamo il nome d'istanza *babelfish*. Poi, una volta aperto il pannello *Components*, trasciniamo due componenti *TextInput* sullo stage, prelevandoli dall'elenco *UI Components:* al primo diamo come nome istanza da_tradurre e al secondo *quale_lingua*. Infine, trasciniamo anche una *TextArea* sullo stage e diamole come nome di istanza *traduzione*.**

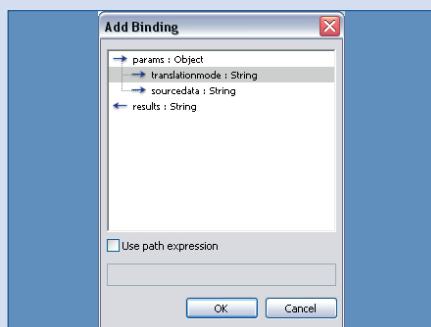
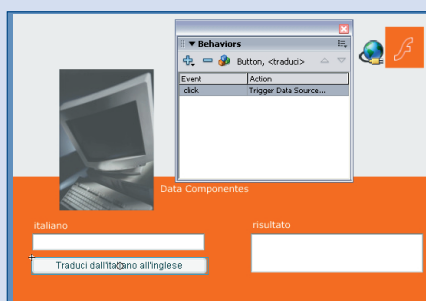
complessiva del prodotto non si discutono; peccato per la documentazione, che spesso appare incompleta. Quando si cercano informazioni su un nuovo componente o su una nuova funzione utilizzando l'help panel, puntualmente appare una scritta che ci invita a cercare aggiornamenti sul sito, rivelando la totale mancanza di informazioni in merito. Per fortuna, il sito Macromedia ha rivelato nel tempo una vitalità e velocità di aggiornamento dei contenuti unica nel suo campo: basta tenere d'occhio *http://www.macromedia.com* per saperne di più sulle novità non documentate. L'unico appunto che si potrebbe fare alla Macromedia è quello di non aver ancora pensato ad un player, le cui risorse non si appoggino quasi esclusivamente sul processore. Allo stato attuale, infatti, nonostante le evidenti migliorie, è ancora impensabile sviluppare video games in Flash del tipo spara tutto, con simulazioni avanzate di percorsi 3d o simili. Un vero peccato se si pensa al fatto che l'architettura, la completezza e la rigorosità di ActionScript 2.0 non hanno nulla da invidiare al celebre Lingo. È facile prevedere un grosso successo di vendite per la versione MX 2004 *Professional* di Flash, anche perché il livello tecnico degli utenti è cresciuto nel corso degli anni. Sicuramente anche molti web designer, che potrebbero dover utilizzare solo occasionalmente le funzioni aggiuntive, non si faranno spaventare dalla necessità di scrivere qualche riga di codice in più, nè dalla differenza di prezzo tra i due prodotti (appena 200 $ di differenza).

☑ Flash MX 2004 Professional

Produttore: Macromedia
Sul web: www.macromedia.it
Software su CD: flashmx2004_trial_en_win.zip
Codice su CD e Web: TutorialFlash.zip

☑ REQUISITI

Per Windows
600 MHz Intel Pentium III o equivalenti
Windows 98 SE, Windows 2000, o Windows XP
128 MB RAM (256 MB raccomanadati)
275 MB di memoria libera sull'hard disk

Per Macintosh
500 MHz PowerPC G3 processor
Mac OS X 10.2.6
128 MB RAM (256 MB raccomanadati)
215 MB di memoria libera sull'hard disk

3 **Sullo stage, dopo aver selezionato il componente *WebServiceConnector*, apriamo il pannello *Component Inspector*, e scegliamo la scheda *Bindings*, attraverso la quale decideremo il tipo dei dati e indicheremo quali componenti trasmettono informazioni al web service e quali le ricevono. Clicchiamo sul pulsante *Add Binding*; selezioniamo *translationMode:String* e poi clicchiamo sul pulsante *Ok*. Nel pannelo *Component Inspector*, facciamo doppio clic sul campo *bound to*: così facendo si aprirà la finestra di dialogo omonima. Selezioniamo *TextInput <quale_lingua>* e premiamo *Ok*. Sempre dal pannello *Component Inspector*, premiamo nuovamente il tasto *Add Binding* e scegliamo *sourcedata: String* e poi *Ok*. Anche questa volta dobbiamo effettuare un doppio clic sul campo *bound to*, poi selezionare *TextInput <da tradurre>* e premere su *Ok*. Infine, cliccato nuovamente il tasto *Add Binding*, scegliamo *results : String* e *Ok*. Effettuiamo anche questa volta un doppio clic sul campo *bound to* e, nella finestra dialogo, scegliamo *Text Area <traduzione>*.**

4 **A questo punto possiamo rifinire il tutto, inserendo un pulsante che attivi il traduttore. Dal pannello *Components*, trasciniamo sullo stage l'istanza di un Button e diamole come nome istanza traduci. Nel pannello *Component inspector*, selezioniamo la scheda *Parameters* e scriviamo "*traduci dall'italiano all'inglese*". Questa scritta apparirà sul pulsante. Dal pannello *Behaviors*, scegliamo *Data>Trigger Data Source*. Nella finestra di dialogo che apre, selezioniamo il componente *babelfish* e, infine, clicchiamo su *Ok*. Per concludere, scriviamo nel primo fotogramma *quale_lingua.text="it_en";* questo valore comunica al web service che la traduzione si svolgerà dall'italiano all'inglese - con *en_fr*, ad esempio, avremmo ottenuto una traduzione dall'inglese al francese. Provando il filmato, possiamo notare che funziona perfettamente, a patto che sia attiva una connessione e che il file *swf* si trovi sul nostro computer e non su un server remoto. Questa limitazione deriva dalle restizioni attuate dalla Macromedia in materia di scambio di dati tra domini diversi. In pratica, un filmato Flash che viene riprodotto all'interno di un browser web, non ha accesso a dati che risiedano fuori dal dominio su cui si trova. Un discorso che si applica anche su uno stesso server: ad esempio, un filmato situato in un sottodominio non può leggere dati dal dominio "*genitore*", e viceversa.**
Come fare per ovviare al problema, senza ricorrere a linguaggi lato server, che facciano da intermediari? La risposta della Macromedia si chiama *policy file*: nello scambio di dati tra domini, chi mette a disposizione il web service, per consentire il libero utilizzo del servizio, dovrebbe predisporre un file di sicurezza sul proprio server *(cross-domain policy file)*. Un "*policy file*" è un semplice file XML, che, una volta posto sul server, comunica al Flash Player di consentire l'accesso diretto ai dati su quel server, senza avvisare l'utente che l'accesso viene consentito. Se un server pubblico ha un *policy file*, tutti i filmati Flash possono accedere ai suoi dati.
Questo file viene salvato come *crossdomain.xml* e posizionato sulla root del public server.
Per fare un esempio il contenuto del file xml dovrebbe essere di questo tipo:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com
        /xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
  <allow-access-from domain="*" />
</cross-domain-policy>
```

DarkBASIC Professional

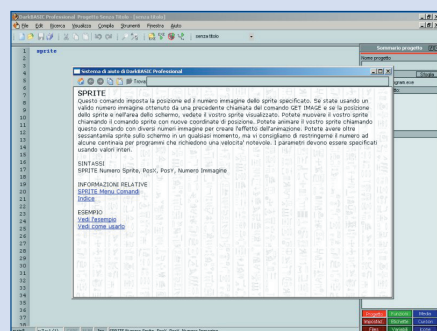
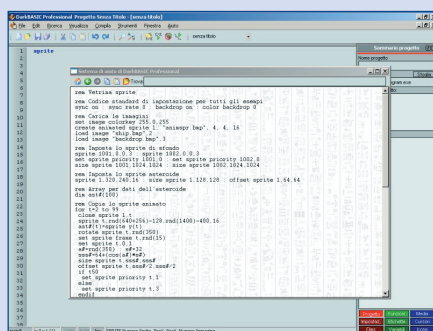
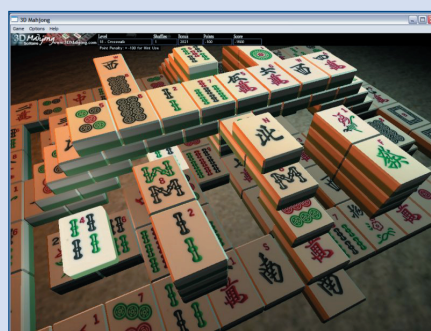
Per creare videogiochi ed applicazioni multimediali di livello professionale, con la semplicità garantita dal Basic.

I videogiochi hanno da sempre affascinato grandi e piccoli utenti di computers, dai tempi di Pong a quelli attuali di Half Life, ed è molto difficile per chiunque ammettere di non aver mai giocato con un computer, e magari di non aver desiderato di creare un videogioco di successo, ma per molti quest'ultimo è rimasto solo un desiderio, perché il C++ o l'Assembler non erano linguaggi semplici da utilizzare. Adesso potete finalmente realizzare il vostro desiderio utilizzando DarkBASIC Professional, il linguaggio di programmazione creato da The Game Creators, la cui caratteristica più importante è la possibilità di scrivere codice per un videogioco o realizzare un'applicazione multimediale usando un linguaggio, semplice quale il BASIC ma molto potente, che sfrutta pienamente l'hardware attuale. Realizzare un videogioco con DarkBASIC Professional non è più un'impresa riservata a programmatori con anni di studi e di programmazione alle spalle, ma è alla portata di tutti, anche di un singolo. Come avviene tutto questo?

Fig. 1: *L'IDE integrato con due esempi di giochi forniti con il linguaggio.*

UN LINGUAGGIO SEMPLICE...

Appena si avvia DarkBASIC Professional, abbiamo la possibilità di scrivere il gioco in un ambiente IDE veramente notevole, con uno degli editors di codice più flessibili mai provati, in cui tutte le linee sono numerate e ogni comando viene evidenziato con un colore. Inoltre, muovendo il mouse al di sopra di un comando viene visualizzato un suggerimento che mostra i parametri da fornire. Il pulsante

ALL'OPERA CON DARKBASIC PROFESSIONAL

1 L'editor in azione. Premendo *"F1"* mentre si scrive il comando *sprite,* si apre una finestra che descrive cosa fa il comando. E' possibile visualizzare un esempio...

2 ... nella stessa finestra d'aiuto per esaminare da vicino la sintassi, ed eventualmente caricarlo al posto del sorgente per osservare quello che si può fare con i comandi sugli *sprites.*

3 Con DarkBASIC Professional è possibile realizzare videogiochi commerciali di successo tipo *3D Mahjong,* un popolare gioco di carte presente su *www.3dmahjong.com ...*

di debug permette di eseguire il codice in modalità passo-passo, o rallentarne l'esecuzione in modo da seguire lo sviluppo dei vostri dati. Nello stesso tempo una guida in linea completa di sorgenti di esempio, liste di comandi e collegamenti a progetti di esempio, è sempre disponibile sullo schermo per una spiegazione dei comandi, per copiare ed incollare, o per aiutarvi nella ricerca di errori. È un IDE perfetto sia per nuovi utenti che per programmatori veterani. Creare oggetti grafici è veramente semplice. Volete disegnare un cubo, una sfera? Allora usate i seguenti comandi:

```
Make Object Cube 1,100
```

e disegnate un cubo di dimensioni 100 e di colore bianco per default mentre:

```
Make Object Sphere 1,100
```

disegna invece una sfera bianca di dimensioni 100. Volete fare qualcosa leggermente più complessa come disegnare un cubo, colorarlo e poi muovervi attorno con la cinepresa?
Ecco il codice che fa tutto questo:

```
make object cube 1,100
color object 1,rgb(255,128,255)
do
 control camera using arrowkeys 0,1,1
loop
```

... E POTENTE...

Potete ovviamente realizzare codice più avanzato degli esempi suddetti, importare modelli 3D nei vostri giochi, in formati diversi, compresi quelli di giochi famosi come Quake e Half Life, applicare textures, effetti pixel e vertex shaders, controllare ogni membro dell'oggetto, creare e gestire matrici per ampi paesaggi…e tanto altro ancora, come illustrato nelle caratteristiche del linguaggio, ma se ancora non foste convinti del linguaggio, perché non provate a sentire e scambiare informazioni con altri utenti di DarkBASIC Professional in Italia?

... IN ITALIANO!

A differenza di altri linguaggi o tool di programmazione per creare videogiochi, DarkBASIC Professional è disponibile in italiano ed è presente nel nostro Paese grazie al sito ufficiale italiano: *http://www.kataxia.com*, gestito da

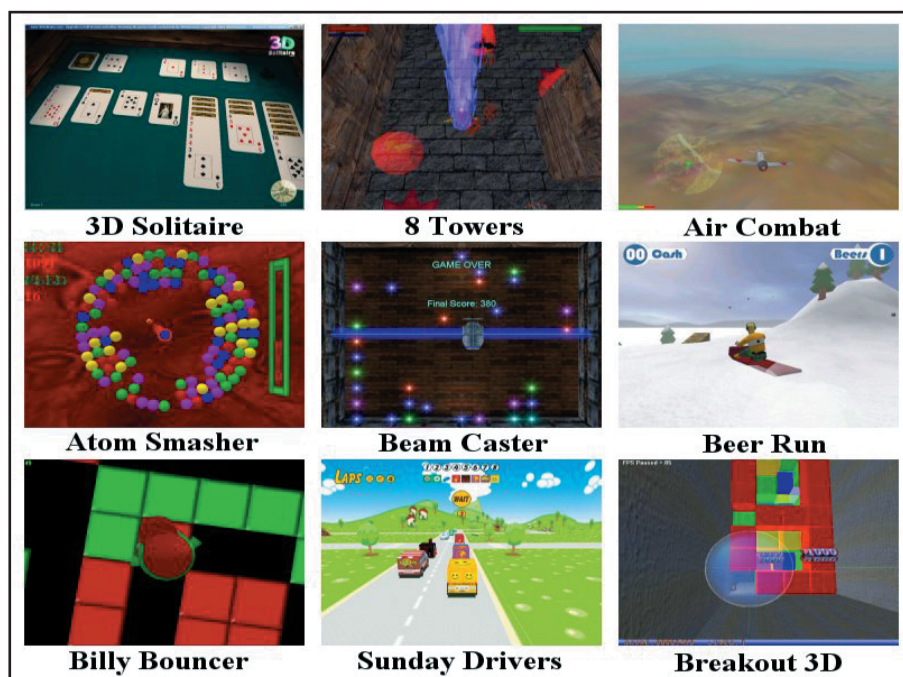



Fig. 2: Con DarkBASIC Professional è possibile creare tantissimi tipi di giochi.

4 ... oppure creare una galleria d'arte virtuale, come nell'esempio di un programma realizzato da un utente americano, con possibilità di aggiungere i vostri capolavori...

5 ... o ancora altre applicazioni multimediali come VsdTracker per riprodurre fedelmente la vostra musica con DarkBASIC Professional!

appassionati programmatori di videogiochi, dove possono incontrarsi sia il neofita che l'utente esperto, nel cui forum è possibile chiedere informazioni se qualche argomento del linguaggio non è chiaro e dove è possibile gratuitamente prelevare gli aggiornamenti di DarkBASIC Professional.

CONCLUSIONI

Il processo di creazione di videogiochi diventa divertente, facile e non troppo gravoso economicamente, quindi accessibile a tutti coloro vogliano provarci. DarkBASIC Professional è il più avanzato applicativo per sviluppare giochi attualmente in commercio, costruito sullo schema del BASIC. Difficilmente troverete programmi che rendono così facile incorporare tutte le speciali caratteristiche ed effetti che vedete nei giochi attuali e che offrono i benefici della tecnologia Microsoft DirectX 9. Il termine professional serve a differenziare questo linguaggio da DarkBASIC, precedente prodotto di The Game Creators ltd, del quale ha mantenuto tutti i benefici aggiungendo nuovi tipi di dati, un motore grafico 3d avanzato, l'accesso a basso livello per gli oggetti e tantissimi nuovi comandi. Nella versione ufficiale non viene fornito un editor di livelli, ma è facilmente possibile reperire su internet editor, anche gratuiti, realizzati da altri utenti del linguaggio. Infine non dimenticate di osservare queste pagine perché a breve saranno disponibili dei mini tutorial sul linguaggio con esempi commentati dalle basi fino alla creazione di un gioco completo.

Roberto Lo Storto

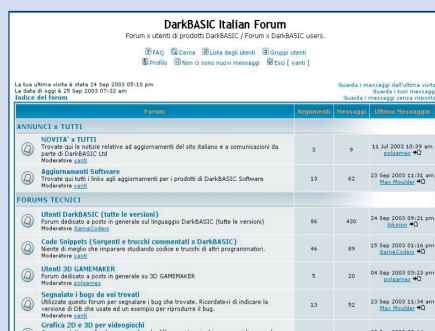
Fig. 3: *Un esempio "reale" di Bone Animation.*

☑ DarkBASIC Professional

Produttore: The Game Creators LTD (Regno Unito)
Sul Web: www.thegamecreators.com
In Italia: www.kataxia.com
Nel CD: \soft\tools\dbprodemo

Tra le caratteristiche del linguaggio

- **Per tutti i giochi creati con DarkBASIC Professional non c'è bisogno di pagare royalties o licenze particolari.**
- **Tutti i giochi possono essere distribuiti come eseguibili EXE indipendenti.**
- **Gestione BSP**
- **Insieme Visibilità Potenziale**
- **Pixel & Vertex Shaders**
- **Ombre in tempo reale**
- **Riflessi realistici**
- **Luci**
- **Matrici**
- **Terreno Avanzato**
- **Cineprese multiple**
- **Sistema Particellare**
- **Gestione sprites 2D (trasparenza, rotazione, sfumatura, animazioni, ...)**
- **Gestione collisioni**
- **Mappatura con rilievo (Bump Mapping)**
- **Mappatura di luce (Light Mapping)**
- **Oggetti con multi tessiture**
- **Animazioni con scheletro (Bone based animations)**
- **Cartoon Shading**
- **Accesso a basso livello dei dati dell'oggetto**
- **Supporto multigiocatore**
- **Manipolazione di vettori**
- **Uso di DLL esterne per espandere il linguaggio con nuove funzioni**
- **Compilatore con possibilità di criptare e comprimere gli eseguibili creati col 100% di codice macchina**
- **Debugger integrato**
- **Editor con evidenziazione sintassi e guida in linea.**

ESPLORARNE LE POTENZIALITÀ

1 **Visitate il sito italiano dedicato a DarkBASIC Professional: *www.kataxia.com* per confrontarvi con altri utenti del linguaggio e aumentare le vostre conoscenze.**

2 **Entrate nella Galleria ed ammirate alcune creazioni oppure cercate informazioni su programmi utili tra cui l'editor di livelli gratuito *"Programmer's Heaven"...***

3 **...o il più famoso *MatEdit* e partite alla grande creando livelli per il vostro gioco con DarkBASIC Professional in cui l'unico limite è la vostra immaginazione.**

DBDesigner 4

Progettazione di database relazionali, ottimizzato per lavorare in coppia con MySQL.

DBDesigner 4 è uno strumento davvero completo e versatile, capace di dare un concreto aiuto nello sviluppo di database complessi. La funzionalità di questo software non è solo quella di assistere il progettista nella stesura di diagrammi ER *(Entità-Relazione)*, ma anche di interagire con il database server. Questo ci consente, ad esempio, di aggiornare lo schema delle tabelle, oppure effettuare il reverse engineering di un database esistente. Già al primo utilizzo, DBDesigner dà l'impressione di un programma completo e professionale oltre che veramente semplice da usare. Infatti DBDesigner 4 dispone di un'interfaccia utente ben realizzata, in linea con i moderni strumenti di sviluppo software. In alto troviamo la classica barra dei menu, a sinistra la palette di strumenti e funzioni più frequentemente utilizzati e a destra un contenitore con 3 pannelli:

Navigator & Info: per avere informazioni sul documento e selezionare la porzione dello schema da visualizzare;
Datatypes: permette di visualizzare e gestire i tipi di dati disponibili;
DB Model: mostra le tabelle presenti nel progetto, con i dettagli relative alle colonne ed alle relazioni;

La zona centrale, più ampia, rappresenta la nostra area di lavoro. E' qui che andremo a costruire il diagramma per il database da progettare, inserendo nuove tabelle, definendone le tabelle e le relazioni con il resto del progetto.

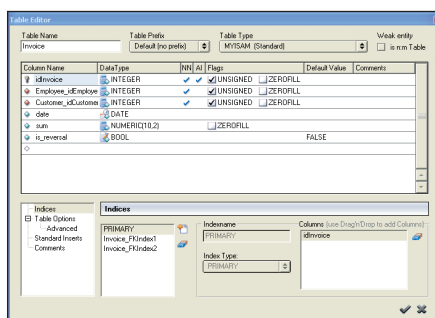



Fig. 2: *L'editor delle tabelle ci consente di definirne tutte le caratteristiche.*

FUNZIONALITÀ

Come altri strumenti per la progettazione di basi di dati, una volta terminata la costruzione del modello, DBDesigner 4 ci permetterà di ottenere il listato SQL necessario a creare il database progettato. Inoltre sarà possibile esportare l'immagine, in formato PNG o BMP, del modello realizzato, molto utile per esaminare il proprio progetto "su carta". DBDesigner è dotato di due funzionalità aggiuntive:

Database Synchronisation
permette di semplificare l'aggiornamento del database, attraverso la connessione direttamente al server. In pratica, piuttosto che utilizzare ripetutamente la generazione di script SQL, DBDesigner confronta la struttura delle tabelle presenti nel modello con quelle definite nel database, modificando queste ultime in modo che coincidano con quelle del modello progettato;
Reverse Engineering
sfruttando la tecnica di ingegneria inversa, DBDesigner è capace di collegarsi ad un database server, prelevare la struttura di un database e creare un modello in base alle meta-informazioni presenti nell'archivio. Questa possibilità si rivela particolarmente utile quando è necessario lavorare su un database già esistente, e del quale non si hanno i modelli di progetto. Non meno importante la possibilità di importare modelli sviluppati con Erwin, un software prodotto dalla Computer Associates e molto diffuso in ambienti Microsoft Windows.

Fig. 1: *L'area di lavoro di DBDesigner, ricca e curata nei particolari.*

CONCLUSIONI

In conclusione possiamo dire che DBDesigner 4 è senza dubbio un ottimo strumento, capace non solo di semplificare molto il lavoro durante la progettazione di un database, ma di seguirne l'evoluzioni attraverso modifiche ed aggiornamenti, ancor più semplici grazie alla funzione di sincronizzazione. Il mondo del software open source aveva proprio bisogno di un software come DBDesigner 4, soprattutto considerando che alcuni dei database server più utilizzati, come MySQL e PostgreSQL, non forniscono un ambiente grafico per progettazione di database.

☑ DBDesigner 4
Produttore: fabFORCE.net
Sul Web: www.fabforce.net/dbdesigner4/
Prezzo: Gratuito
Nel CD: \soft\tools\DBDesigner.zip

Integrated Performance Primitives 3.0

Dalla Intel®, le migliori librerie per far correre le tue applicazioni: con un framework per il raggiungimento di prestazioni al top.

Le *Intel Integrated Performance Primitives* (Intel® IPP) abbracciano un ampio spettro di problematiche e riescono a dare una spinta decisiva in numerosi campi. Le IPP sono essenzialmente divise in cinque gruppi:

- Audio, JPEG, speech e video codifica/decodifica
- Riconoscimento visivo
- Crittografia
- Processo digitale di immagini e segnali analogici
- Riconoscimento vocale

Le Intel® IPP costituiscono una poderosa libreria che può essere vantaggiosamente utilizzata per ottimizzare le prestazione delle applicazioni che sviluppiamo su qualsiasi processore della famiglia Intel®: attraverso un'unica API che fa da interfaccia a tutte le piattaforme, gli sviluppatori possono inoltre contare su un'ampia compatibilità ed una sicura riduzione dei costi.

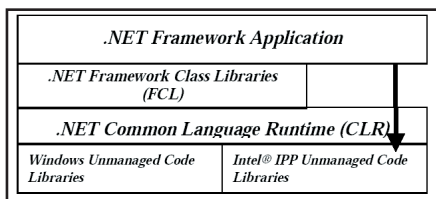



Fig. 1: Sul sito ufficiale della Intel è disponibile un tutorial su come sfruttare le IPP anche su piattaforma .NET.

Fig. 2: Risultati di codifica e decodifica delle immagini in standard MPEG-2.

NUOVE FUNZIONI

Con la versione 3.0 sono stati coperti due nuovi campi: la sintesi vocale e la codifica video. Le Intel® IPP sono dunque particolarmente indicate per la realizzazione di applicazioni di multimediali di livello professionale, in cui il raggiungimento di prestazioni allo stato dell'arte diviene di fondamentale importanza. Le Intel® IPP sono adatte ad essere integrate in qualsiasi progetto C o C++, e particolarmente interessanti risultano gli esempi MFC inclusi nel pacchetto di installazione. Da ricordare inoltre che le librerie IPP sono tutte thread-safe e possono dunque essere impiegate in ambienti threaded senza causare alcun problema. Le librerie IPP sono state utilizzate anche da Macromedia, nello sviluppo del Player Flash: grazie alle IPP è stato possibile ridurre tenere ridotta la dimensione dei filmati Flash, senza andare a discapito della qualità e della fluidità delle prestazioni audio-visive.

DOCUMENTAZIONE

Nel pacchetto di installazione sono inclusi degli ottimi manuali in formato PDF che esplorano in profondità le caratteristiche e gli utilizzi delle librerie. Non lasciatevi spaventare dalle migliaia di pagine che vi troverete davanti: sfogliate l'indice e dirigetevi verso la sezione che fa al caso vostro.

INSTALLAZIONE

Al fine di una corretta installazione, è necessario collegarsi al link *www96.intel.com/cme/showSurv.asp?formID=1474* e completare la form di registrazione, omettendo l'ultimo passo: il download delle librerie, già presenti nel CD allegato ioProgrammo. In pochi istanti riceverete un file di attivazione. Salvate il file in una qualsiasi directory e lanciate il programma di installazione delle librerie. Vi verrà chiesto di indicare la posizione del file di licenza appena ricevuto. Alla fine dell'installazione è possibile lanciare un interessante tutorial che fornisce un'utile overview di questa estesa libreria.

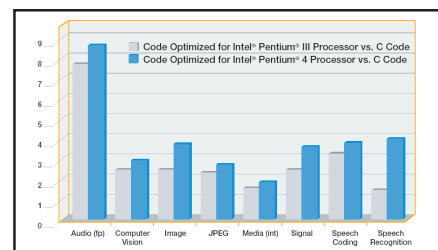



Fig. 3: Guadagno tipico nelle prestazioni rispetto ad applicazioni sviluppate senza fare utilizzo delle Intel IPP.

☑ **Integrated Performance Primitives 3.0**

Rivenditore per l'Italia: Creactive
Sul Web: www.creactive.net/Intel
Prezzo: € 259 ver. commerciale
€ 60 ver. accademica
Nel CD: ipp30win1.exe

Metamill 3.1

UML e reverse engeneering: un ambiente completo per i diagrammi che hanno rivoluzionato lo sviluppo.

La nuova versione di un software per la modellazione software UML completamente visuale. Metamill risulta particolarmente semplice da usare, in special modo se paragonato ad altri software di modellazione UML. Tra le caratteristiche più interessanti c'è il repository, attraverso cui più sviluppatori possono collaborare utilizzando i medesimi elementi senza "pestarsi i piedi". Con Metamill è possibile sviluppare qualsiasi tipo di diagramma UML, inclusi i diagrammi di deployment. I modelli sono salvati in formato XMI (XML Metadata Interchange), cosa che consente un facile riutilizzo ed una più semplice integrazione dei dati così generati in ambienti eterogenei. Una volta sviluppato il modello, Metamill è in grado di produrre il codice relativo sia in C++ sia in Java. E' anche possibile il processo inverso: a partire da codice Java, C++ o C# è possibile risalire alle corrispettive rappresentazioni UML, semplificando le operazioni di reverse engineering e la manutenzione di codice legacy. La documentazione del codice risulta semplificata dalla possibilità di generare automaticamente file HTML di commento. Sviluppato in C++, Metamill si presenta particolarmente veloce. Molto utile, in fine, la possibilità di utilizzare template.
Versione di prova valida trenta giorni.

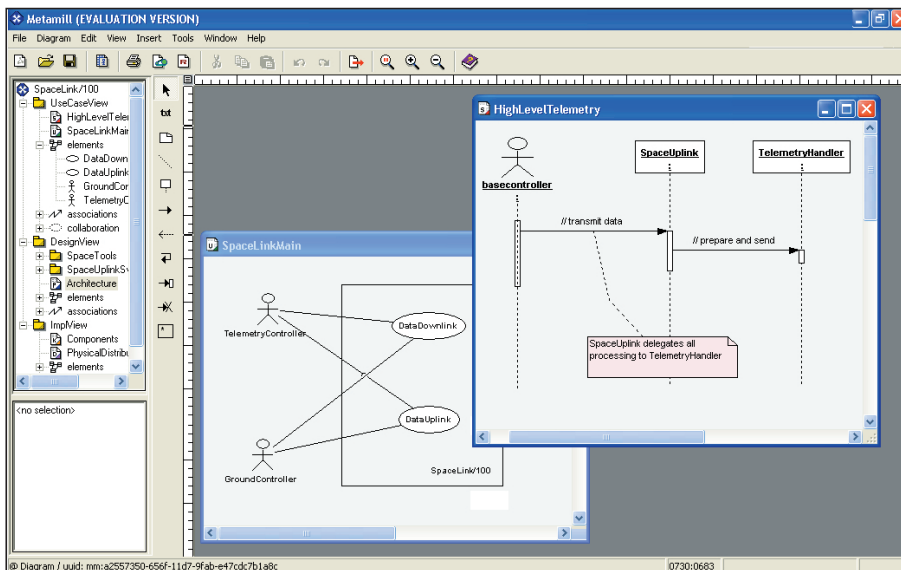



Fig. 1: *Gli use case diagram sono completamente scalabili.*

☑ Metamill 3.1
Produttore: Metamill Software
Sul Web: www.metamill.com
Prezzo: $125
Nel CD: mmill31.exe

AUTHENTIC 2004

Con la semplicità di un world processor, tutte le dotazioni dei migliori ambienti di sviluppo per XML.

Dai creatori di XMLSPY, un nuovo eccellente prodotto per l'interazione con XML. AUTHENTIC permette di creare documenti con la stessa semplicità e con l'immediatezza tipica di un word processor, mentre l'oggetto del nostro lavoro resta un documento XML. Si rivela prezioso in tutte quelle circostanze in cui si vogliano condividere e tenere aggiornati documenti e informazioni di qualsiasi tipo, sia all'interno di un'azienda che sul Web. La semplicità è garantita da una interfaccia completamente WYSIWYG e, grazie al supporto per gli elementi grafici e le tabelle, si rivela come una valida alternativa agli editor HTML. Al momento della prima installazione è necessario cliccare su "FREE Evaluation key" e fornire il proprio indirizzo e-mail: vi sarà inviata la password per l'attivazione del prodotto.

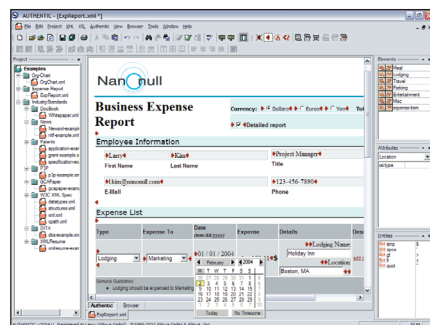



Fig. 1: *Authentic in azione su di una pagina Web.*

☑ AUTHENTIC 2004
Produttore: Altova
Sul Web: www.altova.com
Prezzo: Gratuito
Nel CD: AUTHENTICDeskComplete2004.exe

IntelliJ IDEA Aurora

Una nuova stella nel firmamento Java

Idea è un ambiente di sviluppo che consente di creare applicazioni Java all'interno di un IDE ben fatto e gradevole. Pieno supporto per JSP/EJB, integrazione con Ant e JUnit, supporto per XML e una cospicua collezione di API per la realizzazione di plug-in. Inutile sottolineare che sono presenti tutte le più importanti caratteristiche comuni in un moderno IDE: un ottimo debugger, avanzate funzioni di search&replace, colorazione sintattica. Una menzione particolare la merita il completamento automatico del codice che si fa apprezzare con l'uso per la sua semplicità e per la flessibilità con cui segue le abitudini dello sviluppatore. Al momento dell'installazione è necessario inserire uno username ed una password, per ottenere i quali è presentato un link ad una pagina Web. Una volta compilato il form con i nostri dati, ci verrà inviata una chiave di attivazione valida trenta giorni.

Nel CD: Idea922.exe

Visual Paradigm for UML Community Edition 2.2

Un potente ambiente di sviluppo UML gratuito

Un ambiente di sviluppo integrato per UML (Unified Modeling Language) che farà la gioia di tutte le figure coinvolte nello sviluppo di applicazioni di grandi dimensioni. Completamente gratuito, non rinuncia ad una ottima interfaccia di livello professionale e copre tutte le maggiori esigenze di chi si occupa di UML. Grazie ai numerosi tutorial on-line (*www.visual-paradigm.com*) è possibile imparare ad utilizzarlo ad un buon livello in pochi istanti. Specialmente chi è già esperto di UML, si troverà subito a suo agio, grazie ad una interfaccia che riesce ad essere al contempo completa e semplice da utilizzare. Particolarmente efficace per le operazioni di riverse Engineerng da classi Java verso diagrammi facilmente comprensibili. Sin integra perfettamente nella piattaforma Eclipse, come plug-in. Da provare.

Nel CD: vpuml.exe

TestTrack Pro 6.0.1

Tracciare bug e richieste degli utenti

TestTrack è un tool cross-platform per la catalogazione di bug e la pianificazione dello sviluppo software. Può essere interrogato via browser e si integra perfettamente con Visual Studio 6, Visual Studio .NET e Visual SourceSafe. Permette agli utenti Windows e Macintosh di accedere simultaneamente allo stesso database cross-platform. Inoltre, include ottime funzioni per il filtering, permettendo di creare delle liste di problemi organizzate per priorità. Ottima la possibilità di generare dei report. La notifica via mail di tutti gli aggiornamenti al DB ed il pieno supporto a XML rendono sicuramente interessante questo prodotto. Licenza di valutazione della durata di trenta giorni, è possibile tracciare un massimo di 15 bug.

Nel CD: ttprowininstalldl.exe

XMLSPY 2004 Home Edition

Il Re è tornato...

Uno dei più apprezzati editor XML si presenta con questa nuova versione, che qui presentiamo in licenza home, si rinnova e conferma tutte le sue migliori doti. Già nelle fasi iniziali dell'installazione è possibile decidere il livello di integrazione con Windows, scegliendo se associare alcuni file a XMLSPY e se integrare la voce relativa a XMLSPY nei menu contestuali di Explorer. Alla fine dell'installazione è anche possibile scaricare automaticamente dei alcuni tool aggiuntivi. Al primo avvio ci viene richiesto il nostro nome e un indirizzo mail cui verrà spedita in pochi, istanti, la chiave di attivazione necessaria ad attivare il software per 30 giorni. L'interfaccia è alquanto spartana ma ampiamente personalizzabile, ottimo il supporto ai Web Services.

Nel CD: XMLSPYHomeComplete2004.exe

Decafe Pro 3.9

Risalire al sorgente delle classi Java

Con una interfaccia splendida e intuitiva, uno dei più famosi decompilatori per Java: se siete curiosi di sapere come è stata realizzato un particolare applet o un intera applicazione Java, potete affidarvi a Decafe! Basterà indicare il file .class che ci interessa per veder apparire istantaneamente il codice sorgente. Durante la fase di decompilazione del codice, non è necessario avere installata alcuna versione di Java SDK ne' la virtual machine. Questa nuova versione risolve alcuni piccoli problemi delle precedenti. Versione di prova valida ventuno giorni.

Nel CD: decafe39.zip

RenderX XEP 3.6

Per convertire documenti XML in PDF

Un'applicazione commerciale in grado di effettuare la conversione da documenti XML in PDF o in formato PostScript. Accetta in input dati in formato XML e fogli di stile XSL, il rendere è effettuato proprio come combinazione dei due ingressi. Versione dimostrativa, un watermark è applicato su ogni documento prodotto. XEP offra un valido supporto a XSL FO, grafici SVG e funzioni di link PDF-to-PDF. Piccole correzioni in questa nuova versione.

Nel CD: xep36_trial.zip

Installer2Go 3.1.7

Un sistema facile e gratuito per realizzare pacchetti di installazione

Un tool per la creazione di file autoinstallanti che non impegna lo sviluppatore nel dover imparare l'ennesimo linguaggio di scripting: con una interfaccia che punta tutto sul frag&drop, realizzare pacchetti di installazione diventa un vero gioco da ragazzi. Benché gratuito, Installer2Go va incontro a tutte linee guida per la certificazione WindowsXP. La versione 3.1.7 risolve alcuni piccoli bug e relativi alla scrittura nel registro di Windows e alla gestione di stringhe di grandi dimensioni.

Nel CD: i2g1.exe

Code Warehouse 1.02

Un unico grande archivio per tutto il tuo codice

Progettato allo scopo di rendere più facilmente riutilizzabile il codice che scriviamo, Code Warehouse consente di immagazzinare tutti i nostri snippet in un unico archivio. Code Warehouse si occupa di passare al setaccio le nostre classi o interi progetti, al fine di importarne automaticamente le classi nel database. Le ricerche sul codice immagazzinato possono essere effettuate secondo molteplici criteri: autore, data, nome delle classe o procedura, per parole nel codice o anche in modo specifico all'interno dei commenti presenti nel codice.

Nel CD: codeWarehouse.zip

Iron Speed Designer 1.4

Per generare applicazioni Web su piattaforma .NET

Un ottimo aiuto per chi si trova a sviluppare applicazioni Web su .NET: è possibile sviluppare una completa applicazione three-tier, senza la necessità di scrivere codice. Sofisticate interfacce utente ed una robusta logica di accesso ai dati, sono automaticamente generate da Iron Speed, al programmatore è lasciato da scrivere solo il codice strettamente relativo alla logica applicativa. Scrivere meno codice, oltre a ridurre i tempi di sviluppo, consente di ridurre anche la possibilità di introdurre errori. Versione di

prova valida quindici giorni.
Nel CD: Iron Speed Designer Installer.exe

Intelliwizard Designer 1.2.8

Costruisci il tuo Wizard!

I wizard hanno rappresentato un grande passo avanti nella semplificazione delle procedure di interazione fra gli utenti ed il software: con una semplice serie di passi, utilizzando una interfaccia quanto mai lineare e attraverso delle domande di facile comprensione è possibile guidare l'utente e ricavare tutte le informazioni che ci interessano. Intelliwizard consente di costruire dei Wizard che, pur conservando una grande pulizia e semplicità nell'interfaccia, realizzano al loro interno dei complessi percorsi di scelta. La progettazione avviene in gran parte per via grafica in modo da semplificare e migliorare la fase di sviluppo. Versione dimostrativa.
Nel CD: iwzSuite1.2.8-2003.07.10-min.zip

Tarma Installer 2.64.1357

Per avere pacchetti di installazione in più lingue

Attraverso una interfaccia particolarmente user-friendly, creare pacchetti di installazione con Tarma Installer è un vero gioco! Supporta tutte le piattaforme Windows (95, 98, Me, NT 4, 2000 e XP) e si segnala per numerose caratteristiche positive: piccoli pacchetti di installazione, interfaccia semplice e rapida da utilizzare, funzioni di installazione e disinstallazione intelligenti. E' possibile distribuire programmi, documenti, controlli ActiveX, font TrueType e OpenType, driver per periferiche, aggiornamenti di registro e molto altro ancora. Presenta funzionalità specifiche per la distribuzione sicura delle applicazioni sia via Internet si attraverso CD-ROM.Gratuito.
Nel CD: tin2.exe

KernelDriver 6.1

Testare e creare driver per periferiche

Per la creazione di un nuovo driver è sufficiente installare la periferica (sia essa USB, PCI o ISA),scegliere la voce "Create a new project", selezionare il dispositivo dalla lista che si presenta, testare la periferica attraverso l'interfaccia grafica e definire i parametri aggiuntivi che desideriamo specificare. A questi punto avremo il sorgente del nostro driver pronto per essere compilato e lanciato.
Notevoli le funzioni di monitoraggio del sistema: tutti gli interrupt e le chiamate di sistema sono intercettate in un log che consente una puntuale analisi dello stato e dell'evoluzione del sistema.
Versione di prova valida trenta giorni.
Nel CD: KERDRV.EXE

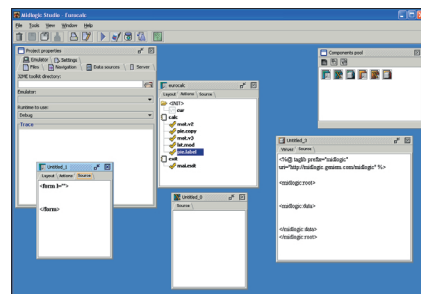
JWin Express Enterprise 2003 Build 385

Da Java ad exe in un click

Velocissimo e semplice da utilizzare, Jwin consente di creare applicazioni Win 32 a partire da qualsiasi applicazione Java. E' necessario avere installato un JDK pari o superiore alla versione 1.4.0. L'interfaccia è strutturata a schede e consente un fine settaggio delle applicazioni che si realizzano. Finalmente uno strumento di livello professionale per la conversione in file eseguibili di progetti Java.
Versione di prova valida cento giorni.
Nel CD: JWinExpressEnterprise-Install.exe

Midlogic Studio 1.1

Sviluppare e distribuire applicazioni J2ME MIDP

Se il buon giorno si vede dal mattino, cominciamo benissimo con un ottimo wizard per l'installazione che si distingue sia per la chiarezza grafica che per la semplicità: tra l'altro, è possibile specificare se utilizzare una delle Virtual Machine esistenti, o installarne una nuova.
Al primo avvio è necessario inserire un chiave di attivazione che si può ottenere attraverso un link presente nello splash. Nella pagina web cui sarete indirizzati, ricordatevi di registrarvi, prima di chiedere la chiave di attivazione, la pagina è poco chiara e i messaggi di errore sono fuorvianti.

L'interfaccia è semplice ed ampiamente personalizzabile, anche progetti di consistenti dimensioni possono essere gestiti agevolmente.
Versione di prova valida trenta giorni.
Nel CD: Install.exe

STYLEVISION 2004

Converte siti HTML in avanzati siti XML-based

Dagli autori di XMLSPY, arriva questa interessante proposta per gli sviluppatori Web, che consente una più agevole migrazione dai tradizionali siti HTML verso più attuali siti completamente basati su XML. STYLEVISION 2004 include una potente funzionalità di import HTML che separa la pagina originale in contenuto XML, foglio di stile XSLT ed un XML Schema.
Nel CD: STYLEVISIONComplete2004.exe

Surround SCM 2.0

Per tenere traccia dei cambiamenti di produzione

L'applicativo garantisce l'accesso al codice sorgente e consente di tenere traccia di tutti i cambiamenti apportati dal team in modo ordinato. Strutturato secondo una logica client/server, l'applicativo può essere utilizzato in un ambiente multipiattaforma: Linux, Mac e Windows. Questa nuova versione prevede la notifica via mail dei cambiamenti, la gestione criptata delle comunicazioni client/server e molto altro ancora. Alla fine della installazione è possibile richiedere la chiave di attivazione necessaria alla corretta utilizzazione dell'applicativo per trenta giorni. Versione dimostrativa valida trenta giorni.
Nel CD: sscmwininstalldl.exe

Color Cop 5.3

Converte valori RGB decimali in triplette esadecimali

Una piccola e utilissima applicazione che consente di ottenere il codice RGB sia in decimale che in esadecimale di qualsiasi colore. Una volta lanciata, la piccola finestra dell'applicazione resta always-on-the-top e, con il classico contagocce (picker), possiamo valutare il colore di qualsiasi punto sullo schermo. Una particolare lente rende anche possibile ingrandire una porzione dello schermo, sempre allo scopo di ottenere i valori dei colori che ci interessano. Gratuito.
Nel CD: colorcop-setup.exe

LTProf 1.4

Profilare applicazioni

Un piccolo ma efficace tool per la profilazione delle applicazioni in relazione al tempo di CPU utilizzato. Attraverso un semplicissimo wizard è possibile specificare il processo da monitorare, dopo di che potremo utilizzare comunemente la nostra applicazione e verificare a posteriori il peso, in termini di tempo-cpu, delle varie operazioni effettuate. Versione di prova valida ventuno giorni.
Nel CD: LTProf_setup.exe

Metodi predittivi di inserimento testi

La funzione T9 dei cellulari

Ai tradizionali sistemi di battitura multipla, per la composizione su tastiere con un numero "limitato" di pulsanti, come quella di un telefono cellulare, si sono succeduti sistemi avanzati di inserimento predittivo; T9, tra questi, è il più conosciuto.

L'esponenziale sviluppo tecnologico degli ultimi anni ha portato con se una serie di benefici per l'utente finale. Tra questi la possibilità di disporre di strumenti sempre meno ingombranti e ovviamente più pratici. Inutile ricordare che il primo elaboratore messo a punto da John Von Neumann occupava con le sue valvole un intero appartamento, mentre oggi a "soli" 50 anni da allora, possiamo servirci di calcolatori miliardi di volte più potenti nel solo spazio del palmo di una mano. La costruzione di hardware su scale di spazio sempre minori è conosciuta come miniaturizzazione. Tale processo, ci consente oggi di disporre, ad esempio, di telefoni cellulari che possono essere comodamente manipolati con una mano e che hanno ingombro minimo, comparabile ad un pacchetto di caramelle. Sembra, che il trend futuro continui a riservarci dimensioni sempre più piccole, si parla di cellulari come orologi o della sola grandezza di un dente. Ma questa estremizzazione delle dimensioni verso valori sempre minori deve fare i conti con delle esigenze pratiche. Sempre con riferimento ai telefoni cellulari, qualcuno si potrebbe chiedere come sia possibile conciliare le dimensioni ridotte all'espletazione di alcune funzioni, come ad esempio la composizione di brevi messaggi di testo (sms). In qualche modo si deve disporre di una tastiera se pur piccola, a meno di usufruire di sistemi di riconoscimento vocale. Certo, penso che sarebbe auspicabile avere a disposizione un set di tasti corrispondenti al completo alfabeto in modo da poter comporre comodamente qualsiasi tipo di messaggio secondo la modalità QWERTY, un tasto una lettera; ma tale scelta si contrappone chiaramente all'esigenza di risparmiare spazio. In effetti, esistono in commercio cellulari con l'intera tastiera, ma sono poco diffusi; il rifiuto del mercato per prodotti che superano alcune dimensioni è una riprova del trend che porta verso il piccolo. Si deve, quindi, poter produrre testi con keyboard ridotte, in particolare i dieci tasti (come si vedrà in realtà sono otto o nove) su cui sono impressi i numeri per la normale composizione telefonica devono essere sfruttati anche per comporre testi. Uno stesso tasto ha molteplici funzioni ed identifica quindi cifre come set di caratteri. T9 è una tecnica di inserimento predittivo di testi che rende "indolore" il processo di miniaturizzazione applicato ai telefoni cellulari fornendo buone prestazioni a fronte di un hardware che per ragioni contingenti di spazio deve essere minimale. Tali tecniche hanno vasti campi di applicazione, non si riducono, infatti, ad essere usate per la gestione di sms su telefoni cellulari, ma hanno un massiccio impiego anche su computer palmari, insomma, in tutte quelle occasioni in cui per ragioni di spazio si è costretti a fare i conti con strumenti di piccole dimensioni e con pochi elementi di comando. Cerchiamo quindi di capire come tali tecniche funzionino e si possano programmare.

☑ T9

Il T9 è il sistema di scrittura SMS che semplifica e velocizza la creazione dei messaggi. Invece di utilizzare il metodo tradizionale, che consiste nel premere i tasti più di una volta per selezionare le lettere più poste in profondità, il T9 si basa su un sistema intelligente; basta che l'utente prema i tasti una volta sola e il cellulare confronterà le lettere con le parole esistenti nel proprio dizionario interno. Per esempio, immaginiamo di voler scrivere la parola "CIAO".

Col metodo tradizionale premeremo i seguenti tasti: [2 abc] [2 abc] [2 abc] [4 ghi] [4 ghi] [4 ghi] [2 abc] [6 mno] [6 mno] [6 mno] Per un totale di 10 tasti.

Col T9 sarà sufficiente la seguente combinazione: [2 abc] [4 ghi] [2 abc] [6 mno] Con un risparmio del 60%.

DA MULTI-TAPPING A T9

Il sogno di chi compone testi è quello di impiegare il meno tempo possibile per la produzione degli stessi. Sistemi di riconoscimento vocale che siano efficienti sono auspicabili per la rapidità e la comodità d'uso. Ma anche la tradizionale tastiera QWERTY (che abbia la conosciuta disposizione dattilografica dei tasti), con la quale sto scrivendo parola dopo parola il presente articolo, non è assolutamente un metodo da disprezzare, specialmente per chi ha preso dimestichezza con la

Fig. 1: ***La disposizione delle lettere sulla tastiera dei telefonini segue l'ordine alfabetico.***

keyboard e riesce a scrivere usando il maggior numero di dita. Tutti sistemi inutilizzabili per maggior parte dei cellulari e per molti palmari. Avendo a che fare con un numero ridotto di tasti, si sono introdotti metodi alternativi per la scrittura del testo. Il primo e più conosciuto tra questi metodi ha il nome di multi-tapping, ossia, battitura ripetuta. Utilizzando tale tecnica per scrivere una singola lettera, bisogna battere più volte lo stesso tasto a seconda dell'ordine in cui essa compare sullo stesso. Così, per scrivere il carattere *a* basterà battere una volta il pulsante 2, mentre la *s* si otterrà con la pressione ripetuta di 4 volte del tasto 7. Non si possono quindi pretendere accettabili performance se si pensa che, per la produzione di una lettera, bisogna in generale, fare più battiture, ed anche in considerazione del fatto che bisogna attendere un tempo, se pur minimo, per la generazione di un singolo carattere, che deve trascorrere interamente qualora i due caratteri da produrre adiacenti appartengano allo stesso tasto. Ad esempio, se si deve scrivere la parola *nonno* bisogna battere il tasto 6 per una volta per scrivere *n,* poi attendere che venga prodotta la lettera e di nuovo battere per due volte il tasto 6, quindi ancora attendere il tempo di produzione e così, continuare digitando ripetutamente sempre lo stesso bottone. Una valida evoluzione è stata introdotta dalla Tegic Communications che ha realizzato il sistema predittivo di inserimento di testo *T9*, la cui sigla significa *"Text on 9 keys"*.

LA FUNZIONE T9

Chiunque possieda un cellulare di recente (ultimi due anni) acquisto di fatto usufruisce delle funzionalità T9. Diamone comunque un rapida quanto completa descrizione. Fermo restando che si fa riferimento ad una tradizionale tastiera (Fig. 1), nel caso dei telefoni cellulari, per produrre una parola di *n* lettere basterà digitare gli *n* tasti corrispondenti alle singole lettere, senza curarsi della posizione che queste hanno sulle cifre associate. Le lettere sono associate alle cifre che vanno da 2 a 9. Il tasto 1 (oppure zero, a seconda dei modelli) produce lo spazio. Quindi per scrivere *nonno*, basterà digitare per 5 volte il tasto 6 e non 7 come nel caso multi-tapping, e non bisognerà ad ogni passo attendere il tempo di produzione tipico della battitura multipla.
Il sistema, in automatico, è in grado di stabilire che quella combinazione di tasti può produrre la sola parola *nonno*.
Ovviamente, può accadere, soprattutto per parole brevi, che uguali combinazioni di tasti, possano essere associate a distinte parole.
Ad esempio, se si digitano i due tasti 64 appare la parola *in*, ma altre parole corrispondono a tale sequenza che si possono opportunamente selezionare a cui daremo il nome di *pseudo-omonimi*. Nel caso specifico, gli altri termini sono: *io, ho, go, im, gn*. Si può notare la presenza del termine inglese *go* che c'è sicuramente perché di uso comune anche nelle comunicazioni in italiano. Inoltre, si scorgono due sigle di cui onestamente ignoravo l'esistenza. La tecnica sviluppata in casa Tegic Communication, quindi, prevede un dizionario che può essere ampliato con nuovi termini. Infatti, se si digita una parola che non è riconosciuta essa si può aggiungere, è il caso di nomi o sigle desuete.
Inoltre, interessante è la caratteristica che dispone le diverse parole secondo un ordine interno che tiene conto della "popolarità" di una parola, per cui nell'esempio esaminato, *in* è preferito a *io* che a sua volta precede il verbo avere in prima persona *ho*. Alcuni modelli di cellulari rendono la composizione ancora più semplice associando un pad o qualcosa di simile (il sony cmdz5 ha una rotella chiamata jog dial) per navigare più rapidamente tra i pseudo-omonimi; cosicché l'uso associato della tastiera e del pad rende la composizione a livelli di rapidità accettabili. È sempre possibile passare (switchare) tra le due modalità, T9 e multi-tapping.
Modelli di cellulari di ultima generazione prevedono sistemi multilingua consentendo

☑ TEGIC COMMUNICATIONS

Il sistema T9: "text on 9 keys" è stato sviluppato dalla Tegic Communication nel 1999, in pochi mesi ha conquistato la fiducia della totalità dei produttori di telefoni cellulari. Si pensi che oggi è presente praticamente su tutti i telefoni cellulari di nuova generazione. Nello stesso anno (1 dicembre 1999) la Tegic è stata assorbita dalla AOL, il colosso delle comunicazioni e internet americano, con la quale collaborava già da prima sul progetto "Instant Messenger". Se si digita www.t9.com **si viene ospitati di fatto in casa AOL. Attualmente il sistema è sviluppato per 42 lingue.**

così la veloce traduzione e una composizione più particolareggiata.
Il sistema T9 è stato introdotto anche su alcuni computer palmari. Ma è arrivato il momento di dedicare la nostra attenzione all'aspetto che maggiormente ci intriga e per cui, noi programmatori, siamo più naturalmente orientati.
Vedremo come si possa strutturare un sistema predittivo di inserimento di testi.

SVILUPPO DI UN PROGETTO

In questo paragrafo tracceremo i passi fondamentali per lo sviluppo di un'applicazione che implementi un sistema predittivo di inserimento di testo, come T9. In questo percorso ci toccherà fare i conti con l'hardware disponibile, inoltre, sarà piacevole fare riferimento ad algoritmi sviluppati tra queste pagine per altri problemi.
Il riferimento è al numero 65 della rivista, quando abbiamo manipolato le parole, per produrre anagrammi. Noteremo che, molte delle esperienze maturate in quella occasione, sono fondamentali per la risoluzione del nostro problema. Un'applicazione che predisponga il dispositivo all'uso del T9, o comunque di un sistema automatico di inserimento testo, deve prevedere la creazione di una hash table che fornisca una corrispondenza tra un numero e una parola o set di parole. Si tratta, quindi, di associare, al numero digitato sulla tastiera del cellulare, la parola o un insieme di parole.
La sequenza di macro operazioni da compiere è riportata di seguito:

```
// Preliminari
download(sorgente, firme)
sort(firme)
union(firme)
load(firme,hashtable)
```

Il file *sorgente* è il dizionario, contenente la sola lista di parole della lingua, ad esempio quella italiana, più le parole di uso comune, anche se non di senso compiuto, quelle che per capirci non sono riportate nel vocabolario della lingua italiana, ma che vengono usate ogni giorno. Tale file è scaricato su una struttura dati in cui si producono una serie di coppie di dati, parola e firma. La firma altro non è che il numero da digitare per produrre la specifica parola.

☑ ALCUNI LIMITI T9

Sono stati realizzati molti studi, numerosi dei quali basati su parametri statistici, per verificare la bontà e la qualità del metodo T9 e di altri simili. In T9 è stata riscontrata una certa difficoltà nella gestione di tutti quei simboli che sono diversi dalle lettere dell'alfabeto, come numeri e caratteri speciali quali punteggiatura, parentesi e altri. Per testi eterogenei si è costretti spesso a "switchare" tra le modalità T9 multi-tapping. Inoltre, sono state comprovate difficoltà di composizione in testi carichi di slang, per la continua necessità di caricare nuovi termini nel dizionario.

Esempi di tale corrispondenza sono:

nonno 66666
nonna 66662
io 46
pane 7263
cane 2263
…

La firma è la vera chiave che porta alla soluzione, infatti, l'idea sta nell'associare le parole con altri elementi su cui effettuare la ricerca, appunto le firme. Nel caso degli anagrammi, le firme altro non erano che le stesse lettere costituenti la parola ordinate in modo crescente, così uguali firme corrispondevano ad anagrammi.
Nel caso specifico, invece, uguali firme generano pseudo-omonimi. La struttura dati firme può essere definita come una matrice di due colonne e m righe, con m il numero di termini del dizionario (cardinalità del dizionario). Nella prima colonna si collocano le firme mentre, nella seconda, le parole corrispondenti.
Una seconda soluzione, ancora più efficiente, implementa la struttura dati come un array monodimensionale costituito dalle firme, in cui ogni elemento ha una lista a puntatori che contiene un unico nodo, in cui è presente la parola corrispondente. Useremo questa seconda soluzione poiché rende più facile il lavoro successivo.
Una volta generato un array di firme, che definisce peraltro la prima impronta del hash table, bisogna ordinare il vettore per firme. Anche qui si possono seguire sostanzialmente due strade. Le due scelte si distinguono in base al tipo che viene assegnato alla singola firma. In alcune simulazioni del T9, che ho trovato nelle mie ricerche su internet, tale tipo è stato definito come stringa di caratteri, poiché è più semplice la fase di produzione della firma che avviene come una banale concatenazione tra singoli caratteri. In tal caso, però, la struttura ordinata non segue l'ordinamento naturale, essendo dipendente dal codice ascii usato.
Se invece, il tipo elementare è un numerico, l'ordinamento avviene in modo naturale e anche il programma risulta più efficiente. In tal caso non è banale la composizione della firma, bisogna trasformare la stringa ottenuta come sequenza di cifre in un numero, per ogni composizione parziale prodotta. Inoltre, il tipo deve essere un intero molto lungo. Essendo più efficiente consideriamo la se-

conda scelta. Come mostrato nell'algoritmo, si procede con l'ordinamento per firme, realizzato da *sort*. La fase successiva prevede il collassamento della struttura (*union*), ossia, qualora ci siano delle ripetizioni di firma, si uniscono in un'unica riga producendo *append* nella lista a puntatori corrispondente. Per cui, ad esempio, in corrispondenza della firma 64, sarà presente una lista a puntatori di 7 elementi, ognuno dei quali contiene le seguenti parole: *in, io, ho, go, im, gn*. Una volta prodotta la hash table si tratta semplicemente di caricare la struttura sul dispositivo *(load)*.
Un secondo stadio di applicazioni opera sul dispositivo che usa il sistema di inserimento. Bisogna, in questo contesto, garantire alcune importanti funzioni che sono; la ricerca e l'aggiornamento.
Esaminiamo alcune possibilità di ricerca. La ricerca dicotomica garantisce buone performance se si pensa che un dizionario si aggira intorno ai 2000 termini e che dopo il collassamento si riducono ulteriormente. Con una decina di interrogazioni si perviene all'elemento richiesto. A tale proposito, si ricorda che per la ricerca binaria si ha una complessità temporale dell'ordine del logaritmo in base 2 della lunghezza del vettore, *O(lg(n))*.
Ma un ulteriore miglioramento si ottiene considerando che la parola si compone man mano che viene digitata, e che ad ogni stadio della composizione, viene proposta una parola fino a quel momento ottenuta. Così, quando si digita il primo numero, si potrebbe semplicemente cercare tra le sole firme di lunghezza 1 che sono solo otto. Se la ricerca prosegue, vuol dire che si sono battuti più tasti; continuiamo a considerare l'esempio precedente; dopo aver digitato 6 si batte il tasto 4, non è necessario ricercare il numero 64 tra tutti.
Si può restringere il range di ricerca a tutti i numeri che iniziano per 6 e così via.
La seconda ricerca, così descritta, si può efficacemente implementare mantenendo una struttura ad albero in cui ogni nodo presenta otto rami, in questo ultimo caso si velocizza ulteriormente la fase.
Discutiamo una piccola appendice, indicando che un parametro di bontà dell'algoritmo risolutore è la dimensione del programma, che deve essere minima in considerazione che i telefoni cellulari non dispongono di molta memoria. In questa ottica le strutture predisposte garantiscono un consumo minimo di risorse.

CONCLUSIONI

Per completezza bisogna dire che vi sono altri metodi di inserimento automatico di testi, usati soprattutto per computer palmari, i più conosciuti sono *graffiti* e *fitaly*.
Gli affezionati della sezione soluzioni, che tento sempre di animare con nuove e a volte insoliti argomenti, sanno che sono un "patito" degli algoritmi legati alla manipolazione di numeri e parole, intesi come forma primitiva di informazione nelle loro parti costituenti, ossia le cifre e le lettere. Quale migliore occasione, quindi, per presentare un tema che legasse i soggetti da me preferiti, tenuto conto anche del fatto che nel caso specifico si tratta di applicare, almeno in parte, alcuni metodi sviluppati in occasione degli studi di anagrammatica automatica. Infine, come spesso ho fatto tra queste pagine, volevo puntualizzare che gli studi sviluppati qualche mese fa per produrre in modo automatico anagrammi non si riducevano al solo divertimento e l'articolo di oggi ne è una puntuale riprova.
Quindi l'algoritimica va sempre esaminata con occhio attento ed in considerazione che una stessa applicazione, se adattata, può risolvere altre tipologie di problemi.
Vi aspetto per la soluzione di nuovi ed interessanti quesiti.

Fabio Grimaldi

☑ ULTERIORI SVILUPPI

Esistono diversi gruppi di discussione sui sistemi predittivi di inserimento testi. Da alcuni di essi sono sorte interessanti proposte per migliorare le potenzialità ad esempio di T9. Una riguarda la possibilità di rendere dinamica la posizione delle parole, che presentino pseudo-omonimi, nella lista a puntatori in base alla frequenza, ovviamente, le parole più usate secondo una teoria delle code dovrebbero collocarsi in modo opportuno nella lista. Un'altra proposta si riferisce alle parole lunghe che potrebbero essere automaticamente completate, è infatti inutile scrivere molte lettere quando il termine che si può comporre è solo uno. Si auspica anche un maggiore facilità di passaggio tra il sistema T9 e il multi-tapping. Ed infine, molti richiedono una più funzionale integrazione tra la tastiera e altri strumenti come piccoli joystick, pad o simili.

☑ DIZIONARIO

Un dizionario nell'area della programmazione è riconducibile alla mera lista di termini a cui è associato solitamente un qualcosa. Nella visione classica, l'associazione è il significato, in informatica non è sempre così. Interessante è la definizione che viene proposta nel linguaggio python, per cui un dizionario è una struttura simile ad un array che si distingue da esso, perché per gli array le chiavi di accesso sono numeri (sostanzialmente gli indici), mentre per un dizionario sono elementi che possono essere di qualsiasi tipo. Inoltre, nel caso più generale, python descrive il dizionario come una struttura in cui nuovi inserimenti non devono essere necessariamente ordinati.

Riconoscimento e sintesi vocale in pratica

Una segreteria studenti in VB

Le Speech API (SAPI) permettono lo sviluppo di applicazioni basate sulla voce, in particolare sul riconoscimento vocale, e sulla sintesi vocale, vale a dire la trasformazione di testo in parlato.

PULSANTI APPLICAZIONI
I pulsanti "Addestra riconoscimento" e "Seleziona Speech Engine" servono, rispettivamente, per visualizzare la procedura guidata che permette il miglioramento del riconoscimento vocale e per visualizzare un form che permette di selezionare uno SE tra quelli installati nel sistema

In questo articolo realizzeremo un sottosistema della segreteria studenti (dal nome Easy Esame), sfruttando gli OCX messi a disposizione dalla versione 4.0 del Microsoft Speech SDK, che ancora oggi meglio si sposano con Visual Basic.

SAPI SDK 4

Le SAPI, sono un'estensione delle API di Windows che consentono di utilizzare in maniera relativamente semplice la tecnologia vocale in VB, fornendo al programmatore 6 controlli ActiveX:

Per il riconoscimento vocale:

Voice Commands
Direct Speech Recognition

Per trasformare in testo una frase letta sotto dettatura:

Dictation

Per la sintesi vocale (conversione dal testo al parlato):

Voice Text
Direct Speech Synthesis

Per l'uso della tecnologia vocale attraverso il telefono

Speech Telephony

Tutte le Speech Api sono accessibili attraverso l'OLE *Component object model*, ed il modello di programmazione è orientato agli oggetti.

Per quanto riguarda il riconoscimento e la sintesi vocale, l'accesso alle Api avviene a due livelli: un livello alto *(Voice Commands* e *Voice Text)*, in cui è possibile usare semplici comandi per utilizzare da subito la tecnologia vocale nelle applicazioni VB: un livello più basso *(Direct Speech Recognition, Direct Speech Synthesis)* che permette di dialogare direttamente con il motore necessario per la riproduzione vocale, lo Speech Engine *(SE)*.
La versione completa dello Speech sdk, può essere scaricata da internet all'indirizzo *www.microsoft .com/speech/download/old/sdk40a.asp*. Insieme al SAPI SDK 4.0, la Microsoft distribuisce i suoi Speech Engine, necessari affinché un'applicazione funzioni. È possibile scaricare anche una versione leggera delle SAPI (circa 7 mega) che però non contiene gli SE. Analizziamo ora in dettaglio i due componenti che andremo ad utilizzare in questo articolo: *Voice Text e Voice Commands*

SINTESI VOCALE CON VOICE TEXT

Il componente *Voice Text* rende possibile la conversione di un testo qualsiasi al parlato. Per averlo a disposizione nella ToolBox di VB, è sufficiente spuntare dalla finestra di dialogo *Componenti* (si ottiene selezionando la voce di menu *Progetto/Componenti*), la voce *Microsoft Voice Text*.
Disegnando il componente in un form avremo a disposizione il controllo *TextToSpeech1*.
L'uso del componente *Microsoft Voice Text* è molto semplice. Per consentire la pronuncia di una qualsiasi frase al Computer, è sufficiente chiamare il metodo *Speak* della classe *TextToSpeech*, passando come parametro la stringa che deve essere pronunciata (ricordiamo che senza nessun speech engine installato il programma non funziona).

Il codice necessario per pronunciare il classico *"Ciao Mondo"* sarà semplicemente:

```
TextToSpeech1.Speak "Ciao Mondo"
```

È importante sapere che il modello di programmazione è asincrono, in altre parole il controllo del programma passa subito all'istruzione successiva senza attendere che sia terminata la sintesi del testo. Quando la sintesi è completata viene lanciato l'evento *SpeakingDone*.
Quindi, se riscriviamo il codice precedente in:

```
TextToSpeech1.Speak "Ciao Ciao Mondo"
MsgBox "messaggio"
```

Noteremo che il messaggio apparirà a video prima che il testo sia letto per intero.
Utilizzando l'evento *SpeakingDone* invece, possiamo fare in modo che il messaggio appaia soltanto quando l'engine ha finito di parlare:

```
Private Sub TextToSpeech1_SpeakingDone()
MsgBox "messaggio"
End Sub
```

Gli SE inclusi nel pacchetto *full* delle SAPI sono tutti in versione inglese, ma nonostante tutto leggono discretamente anche l'italiano. In ogni caso, la programmazione del TTS non dipende dallo SE usato e quindi dalla lingua, per questo è sufficiente installare uno SE italiano per ottenere una corretta lettura del testo nella nostra lingua. Esiste, in ogni modo, la possibilità di costruirsi uno SE che utilizzi la lingua e addirittura il dialetto che si preferisce, ma per far ciò, purtroppo, non basta VB ma si devono usare le funzioni API pure. Per un corretto riconoscimento vocale di parole della nostra lingua è invece necessario uno SE italiano.

RICONOSCIMENTO VOCALE IN VB

Il *Riconoscimento Vocale* (RV) realizza il sogno di tanti, e vale a dire quello di poter dialogare con il computer in maniera naturale, senza l'uso della tastiera o del mouse. Avendo poi a disposizione un sistema di sintesi, si potrà avere anche l'illusione che il computer comprenda le nostre parole e risponda anch'esso con la sua voce. Sebbene questo scenario (per certi versi apocalittico) è ancora lontano, grazie alle *Speech API* (SAPI), anche gli sviluppatori VB possono contribuire alla sua realizzazione. Il riconoscimento vocale si pone come obiettivo fondamentale il riconoscimento delle parole pronunciate in modo continuo da una persona, attività abbastanza complessa, giacché richiede, oltre ad una buona potenza di calcolo e di memoria, una migliore comprensione dei fattori che caratterizzano il linguaggio naturale nel suo complesso. Prima di analizzare il componente messo a disposizione dalle SAPI affrontiamo un pizzico di teoria.

RICONOSCIMENTO DEL PARLATO

In linea di principio, il riconoscimento del parlato comporta l'acquisizione del segnale vocale, la sua conversione in formato digitale, la separazione delle singole parole e il confronto delle stesse con un insieme di pattern di riferimento preregistrati che contengono l'associazione tra la forma parlata e quella scritta. Come si può facilmente intuire, ci si muove in un campo caratterizzato da un alto grado d'incertezza. Per questo motivo, i sistemi attuali introducono dei vincoli sull'utilizzazione del sistema, che comportano delle semplificazioni nelle varie fasi del processo di riconoscimento, ottenendo delle buone prestazioni a scapito di un restringimento del dominio applicativo.
Vediamone alcune problematiche:

Pronuncia per parole isolate: durante la fase di campionamento è necessario isolare le parole che costituiscono la frase. Questo comporta una analisi spettrale abbastanza sofisticata del segnale d'ingresso. Inoltre, in alcuni contesti, due o più parole sono pronunciate in modo continuo e la loro decomposizione nelle parole costituenti non è di facile soluzione. Per queste difficoltà, alcuni sistemi di riconoscimento impongono la pronuncia per parole isolate.

Dipendenza dal parlante: la codifica della voce è usualmente fatta attraverso dei modelli che tengono conto sia dei parametri d'eccitazione (frequenza fondamentale *(pitch)*, guadagno, ecc.) che delle caratteristiche del tratto vocale. La rappresentazione codificata della voce dipende quindi dalle caratteristiche del parlante. Durante la fase di riconoscimento, bisogna confrontare la parola pronunciata con un insieme di parole preregistrate (vocabolario del linguaggio) prodotte utilizzando un modello vocale con parametri predefiniti. In altri termini, il confronto viene fatto tra la parola pronunciata in ingresso e tra un insieme di parole pronunciate da un parlante le cui caratteristiche vocali sono preregistrate: se il sistema conosce le caratteristiche del tratto vocale dell'utente, le possibilità d'errore diminuiscono. Questo principio viene utilizzato da alcuni sistemi, detti *speaker dependent,* che prevedono una fase iniziale d'addestramento del sistema, al fine di estrarre i para-

☑ SELEZIONE DI UN SE

Per selezionare un particolare SE si deve utilizzare il componente Direct Speech Recognition con i metodi: FindEngine e SelectEngine

☑ SISTEMI DI SINTESI

Un sistema di sintesi della voce è una macchina che, a partire da appropriate rappresentazioni di un evento linguistico (parola, messaggio, insiemi di messaggi, ecc.), è in grado di produrre l'emissione sonora corrispondente. La sintesi della voce consente ad un computer di inviare istruzioni o informazioni all'utente attraverso il parlato.

I metodi di sintesi della voce possono essere suddivisi in due categorie:
Sintesi basata sulla codifica di singole parole di voce umana registrata. Tali frammenti sono opportunamente combinati per la composizione del messaggio parlato (sistemi a vocabolario limitato).
Sintesi basata su regole linguistiche, fonetiche e acustiche (sistemi a vocabolario illimitato).

☑ SISTEMI A VOCABOLARIO LIMITATO E ILLIMITATO

I sistemi di sintesi a "vocabolario limitato" possono riprodurre, come indica la denominazione, un numero limitato, seppure grande, di parole e/o messaggi. Nel primo caso, la tecnologia impiegata è molto semplice, la macchina attua la gestione di un archivio che contiene la registrazione degli eventi linguistici da riprodurre e cioè un insieme di parametri ottenuti codificando il vocabolario da riprodurre. Il messaggio viene poi generato selezionando i vocaboli , giustapponendo, attraverso regole opportune, i corrispondenti parametri, inviandoli ad un circuito, che procedendo in modo concettualmente inverso alla fase di analisi, dà luogo all'emissione sonora desiderata. Le applicazioni di questo tipo di macchina si possono intuire facilmente e sono già abbastanza diffuse: annunciatori per stazioni ed aeroporti, risponditori telefonici, allarmi di vario tipo, giocattoli, etc. Con i sistemi di sintesi da testo a "vocabolario illimitato", è possibile trasformare in messaggio verbale un qualunque testo. Per l'elaborazione del segnale sono impiegati diversi stadi di elaborazione basati su differenti competenze specialistiche: linguistiche, fonetiche, acustiche.

Fig. 1: L'interfaccia dell'applicazione EasyEsame.

metri caratteristici del modello vocale dell'utente attraverso la pronuncia di un insieme predefinito di frasi. Sistemi di questo tipo si differenziano per il numero di parole che devono essere pronunciate durante la fase d'addestramento. La rimozione di questo vincolo (sistemi *speaker independent*) aumenta ovviamente la complessità del sistema: un approccio comunemente adottato è di addestrare il sistema con una gran varietà di parlanti, mentre uno più sofisticato è di individuare le caratteristiche fonetiche invarianti tra parlanti.

Dimensione del vocabolario: la dimensione del vocabolario influenza notevolmente il sistema di riconoscimento. Un vocabolario di grandi dimensioni, sebbene sia auspicabile poiché aumenta l'insieme di parole riconoscibili, introduce un maggior grado d'incertezza a causa di una maggiore probabilità di contenere parole simili rispetto a vocabolari di dimensioni più piccole. Ovviamente questo problema si può manifestare anche per vocabolari di piccole dimensioni. Un secondo problema deriva dal fatto che il tempo di ricerca può aumentare considerevolmente: i sistemi con vocabolari di grandi dimensioni utilizzano delle tecniche di filtraggio che diminuiscono la dimensione dello spazio di ricerca, a scapito di un aumento della possibilità d'errori.

Fig. 2: I controlli da aggiungere al progetto VB.

Grammatica: definisce l'insieme delle sequenze di parole ammissibili nel dominio del riconoscimento. Imporre dei vincoli stringenti sulla grammatica comporta ovviamente una minore probabilità d'errore nel riconoscimento.

Ambiente: la presenza di rumori di fondo, il cambiamento delle caratteristiche del microfono, la differenza di condizioni rispetto alla fase d'introduzione dei modelli fonetici di riferimento, ecc., possono influenzare notevolmente l'accuratezza del riconoscimento. Questi effetti possono essere compensati utilizzando dei microfoni aventi dei buoni rapporti Segnale/Rumore e montati in vicinanza della bocca. L'utilizzazione di un sistema di riconoscimento in congiunzione con il telefono, comporta analoghi problemi a causa degli eventuali disturbi presenti nella linea e della limitata banda di frequenze.

IMPOSTAZIONE DEL SISTEMA DI RICONOSCIMENTO VOCALE

Il componente *Voice Command* rende possibile il riconoscimento vocale. Per averlo a disposizione nella ToolBox di VB, è sufficiente spuntare dalla finestra di dialogo *Componenti*, la voce *Microsoft Voice Commands*. Disegnando il componente in un form avremo a disposizione il controllo *Vcommand1*. Analizziamo, ora, i passi necessari all'impostazione di un sistema di riconoscimento vocale.

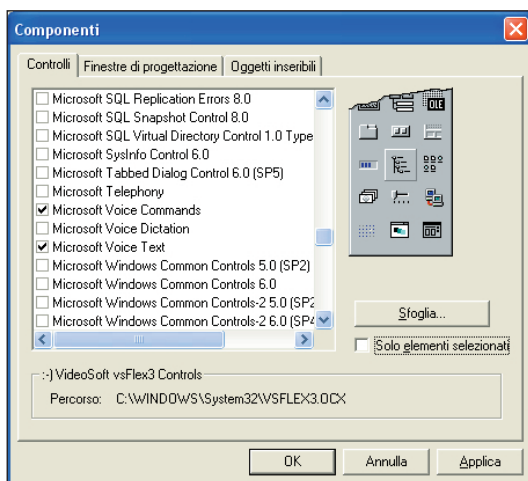
Primo passo: il controllo *VoiceCommand* deve essere inizializzato ponendo ad uno la proprietà *Initialized:*

```
Vcommand1.initialized = 1
```

Il **secondo passo** è quello della definizione del menu dei comandi riconoscibili dal sistema, per questo usiamo la funzione:

```
MenuCreate(Application As String, State As String,
                                    flags As Long) As Long
```

Dove :

Application è il nome dell'applicazione che usa il menu;
State è il nome dello stato d'uso del menu;

flags rappresenta un valore che serve a stabilire in che modo deve essere creato il menu (vedi box a pag. 31). Le costanti usate nelle SAPI sono definite nel modulo *Vbspeech.bas* che pertanto deve essere incluso nel progetto, che restituisce l'identificativo del menu.

Il **terzo passo** è la definizione dei comandi del menu, creato con Menucreate, usando il metodo:

```
AddCommand(Menu As Long, id As Long, command As
        String, description As String, category As String,
                          flags As Long, action As String)
```

Dove:

Menu è l'identificativo del menu di riferimento;
Id rappresenta l'identificativo unico del comando vocale;
Command è la rappresentazione testuale del comando;
Description è una descrizione di quello che fa l'applicazione quando viene lanciato il comando;
Category è la categoria del comando;
Flags indica informazioni sul comando;
Action è una serie di dati inviati all'applicazione quando il comando viene riconosciuto.

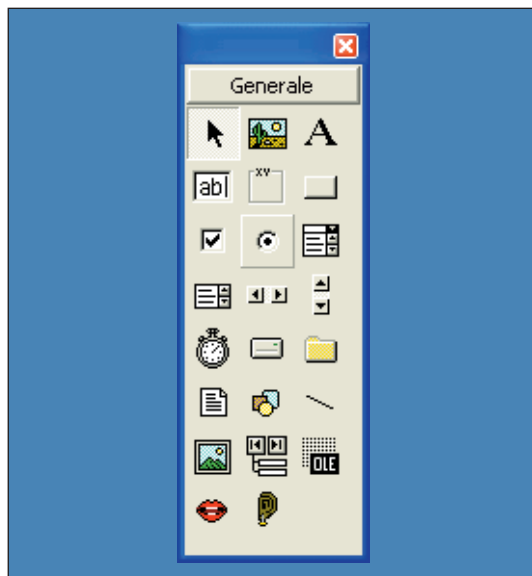



Fig. 2: ***Nella toolbar dei controlli compaiono le icone dei due nuovi componenti installati.***

L'ultimo passo è l'attivazione della fase di ricognizione con il metodo:

```
Activate(Menu as Long)
```

Soltanto quando il menu è attivato lo SE controlla che le parole pronunciate facciano parte dei comandi.
Infine, quando l'applicazione viene chiusa, si deve rilasciare la memoria occupata dal processo di RV con il metodo:

```
ReleaseMenu (Menu as long)
```

STRUTTURA DI EASY ESAME

Scopo di questo articolo è la realizzazione di un'applicazione (chiamata *Easy Esame*) che implementi un sottosistema di una segreteria studenti con il compito di fornire informazioni su:

- Orario corsi
- Programma del corso
- Calendario esami
- Prenotazione esami
- Risultati esami

Le informazioni possono essere memorizzate in un database oppure più semplicemente in file. In *Easy Esame* utilizzeremo la struttura a file poiché l'uso di un database renderebbe necessario descrivere le metodologie di accesso ai database che esulano dallo scopo di questo articolo. Nella directory contenente il programma *(App.Path)* dovranno essere presenti i seguenti file:

Corsi.txt con i nomi dei corsi disponibili - *NomeCorsoOra.txt, NomeCorsoPro.txt, NomeCorsoCal.txt, NomeCorsoPre.txt, NomeCorsoRis.txt* con le informazioni, divise in categorie, di ogni corso. Naturalmente *NomeCorso* individua il nome del corso, perciò per ogni corso disponibile dovranno esistere cinque file. L'interazione con l'utente deve essere il più semplice possibile. Lo studente dovrà selezionare, tramite comando vocale, un corso tra quelli disponibili in una lista mostrata a video. Il nome del corso selezionato deve essere mostrato a video per avere il feedback della corretta scelta effettuata.

A questo punto, sempre tramite comando vocale, lo studente dovrà selezionare il tipo di informazione desiderata, che verrà letta dal sistema (e contemporaneamente mostrata a video). Infine, prima di leggere le ultime due informazioni, *Prenotazioni* e *Risultati*, deve essere richiesto il numero di matricola. Per il disegno dell'applicazione, oltre ai vari fronzoli per abbellire l'interfaccia, saranno necessari:

1 ListBox (ListCorsi) che dovrà contenere l'elenco dei corsi disponibili.

2 TextBox (TextCorso, TextMatricola) che forniranno allo studente il feedback sul corso e sulla

☑ POSSIBILI VALORI DEI FLAG IN MENUCREATE

VCMDMC_CREATE_ALWAYS. Crea un menu vuoto. Se esiste un menu, con lo stesso nome nel database dei menu, viene cancellato e sostituito da quello corrente.
VCMDMC_CREATE_NEW. Crea un menu vuoto. Se esiste un menu con lo stesso nome nel database la funzione restituisce un errore.Il nuovo menu viene memorizzato nel database quando l'oggetto viene rilasciato
VCMDMC_CREATE_TEMP. Crea un menu vuoto. Se esiste un menu con lo stesso nome nel database la funzione restituisce un errore.Il nuovo menu viene memorizzato temporaneamente e viene distrutto quando l'oggetto viene rilasciato
VCMDMC_OPEN_ALWAYS. Apre un menu esistente.Se il menu non esiste ne viene creato uno vuoto.
VCMDMC_OPEN_EXISTING. Apre un menu esistente. Se il menu non esiste la funzione restituisce un errore.

matricola selezionata ed un TextBox *(TextRisultato)* che mostrerà a video l'informazione desiderata.

5 Label che mostrano il nome delle informazioni a disposizione.

1 controllo VoiceCommand.

1 controllo TextToSpeech.

SVILUPPO DI EASY ESAME

A questo punto abbiamo definito tutti gli strumenti necessari per scrivere il codice necessario al corretto funzionamento dell'applicazione.
All'apertura del programma si dovrà:

- **Inizializzare il sistema di riconoscimento**
- **Leggere dal file corsi.txt l'elenco dei corsi disponibili**
- **Visualizzare la lista dei corsi**
- **Inserire nel menu vocale la lista dei corsi**
- **Attivare il sistema di riconoscimento**

Queste fasi saranno implementate nella routine *InitCorsiERiconoscimento*, chiamata nell'evento *Form_Load*. E' necessario, inoltre, definire la variabile *CommandMenu* che dovrà contenere l'identificativo del menu vocale.

```
Option Explicit
Dim CommandMenu As Long
Private Sub Form_Load()
    InitCorsiERiconoscimento
End Sub
```

Vediamo nei dettagli la procedura *InitCorsiERiconoscimento*

```
Private Sub InitCorsiERiconoscimento()
Dim NomeFileCorsi As String
Dim Corso As String
NomeFileCorsi = App.Path + "\corsi.txt" 'definisce il
                                         nome del file da aprire
InitRiconoscimento
Open NomeFileCorsi For Input As #1 ' Apre il file in input.
Do While Not EOF(1)   ' Ripete fino alla fine del file.
   Input #1, Corso 'legge il file una riga per volta
   InserisciInLista Corso
   InserisciInMenuRiconoscimento Corso
Loop
Close #1 'chiude il file
InserisciOpzioniInMenuRiconoscimento
AttivaRiconoscimento
End Sub
```

Dopo aver definito le due variabili *NomeFileCorsi* e *Corso* che conterranno rispettivamente il nome del file da leggere ed il nome del corso in esame, viene chiamata la routine *InitRiconoscimento* che si occupa di inizializzare il sistema di riconoscimento vocale mediante lo schema analizzato in precedenza:

```
Private Sub InitRiconoscimento()
Vcommand1.initialized = 1
CommandMenu = Vcommand1.MenuCreate(
          App.EXEName, "Menu Segreteria Studenti",
                                VCMDMC_CREATE_ALWAYS)
Vcommand1.Enabled = 1
End Sub
```

Le istruzioni necessarie per l'apertura, lettura e chiusura di un file sono commentate all'interno del codice per questo ne tralascio la descrizione. All'interno del ciclo *Do..While*, per ogni corso vengono richiamate due procedure, *InserisciInLista* e *InserisciInMenuRiconoscimento*, che si occupano rispettivamente di riempire la *ListBox* contenente la lista dei corsi, ed inserire il nome del corso nel menu di riconoscimento vocale:

```
Private Sub InserisciInLista(Corso As String)
ListCorsi.AddItem Corso
End Sub
Private Sub InserisciInMenuRiconoscimento(Comando
                                          As String)
Vcommand1.AddCommand CommandMenu, 1,
        Comando, "Corso " + Comando, "Corso", 0, ""
End Sub
```

Infine le ultime due procedure, *InserisciOpzioniInMenuRiconoscimento* e *AttivaRiconoscimento*, si occupano di inserire le possibili opzioni di scelta dello studente nel menu dei comandi ed attivare il sistema di riconoscimento vocale:

```
Private Sub InserisciOpzioniInMenuRiconoscimento()
Vcommand1.AddCommand CommandMenu, 1,
          "calendario", "calendario", "info", 0, ""
Vcommand1.AddCommand CommandMenu, 1,
              "risultato", "risultato", "info", 0, ""
Vcommand1.AddCommand CommandMenu,1,
        "prenotazione", "prenotazione", "info", 0, ""
Vcommand1.AddCommand CommandMenu, 1,
            "programma", "programma", "info", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "orario",
                            "orario", "info", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "fine",
                              "fine", "Fine", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "0",
                         "zero", "matricola", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "1",
                          "uno", "matricola", 0, ""
```

☑ VOICE TELEPHONY CONTROL

Il controllo Voice Telephony (VT) permette di creare applicazioni di telefonia multi-linea con riconoscimento e sintesi vocale e con l'identificazione dei toni DTMF. L'uso della rete telefonica pubblica rende possibile una serie di servizi a mezzo voce, in cui il computer comprende e risponde. VT è un controllo multi-threaded, in cui ogni thread è associato ad una linea telefonica. Naturalmente se si ha a disposizione una sola linea telefonica si avrà un solo thread.

☑ CONFIGURARE L'HARDWARE

Le SAPI forniscono un utile applicazione, Micwiz.exe, per essere sicuri che il microfono sia compatibile con la scheda sonora e per avere una buona regolazione del volume. L'uso di micwiz e abbastanza semplice ed intuitivo; permette di configurare al meglio il microfono seguendo le istruzioni guidate del programma.

```
Vcommand1.AddCommand CommandMenu, 1, "2",
                              "due", "matricola", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "3",
                              "tre", "matricola", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "4",
                              "quattro", "matricola", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "5",
                              "cinque", "matricola", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "6",
                              "sei", "matricola", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "7",
                              "sette", "matricola", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "8",
                              "otto", "matricola", 0, ""
Vcommand1.AddCommand CommandMenu, 1, "9",
                              "nove", "matricola", 0, ""
End Sub
Private Sub AttivaRiconoscimento()
Vcommand1.Activate CommandMenu
End Sub
```

Nel momento in cui l'utente pronuncia un comando tra quelli presenti nel menu, viene lanciato l'evento *CommandRecognize*, perciò in questo evento scriveremo il nucleo del codice di gestione di Easy Esame:

```
Private Sub Vcommand1_CommandRecognize(ByVal ID
   As Long, ByVal CmdName As String, ByVal Flags As
 Long, ByVal Action As String, ByVal NumLists As Long,
 ByVal ListValues As String, ByVal command As String)
Select Case command
   Case "calendario"
      If NessunCorsoSelezionato Then Exit Sub
      EseguiMenuCalendario
   Case "risultato"
      If NessunCorsoSelezionato Then Exit Sub
      If NessunaMatricolaSelezionata Then Exit Sub
      EseguiMenuRisultati
   Case "prenotazione"
      If NessunCorsoSelezionato Then Exit Sub
      If NessunaMatricolaSelezionata Then Exit Sub
      EseguiMenuPrenotazioni
   Case "programma"
      If NessunCorsoSelezionato Then Exit Sub
      EseguiMenuProgramma
   Case "orario"
      If NessunCorsoSelezionato Then Exit Sub
      EseguiMenuOrario
   Case "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"
      TextMatricola = TextMatricola + command
   Case "fine"
       TextToSpeech1.Speak "Grazie per aver usato il
                                              servizio"
       TextMatricola = ""
       TextCorso = ""
   Case Else
      TextCorso = command
End Select
End Sub
```

In pratica, ogni qualvolta l'utente pronuncia una parola che appartiene al set dei comandi predefiniti, viene effettuato un *Select Case* sul comando. Se l'utente pronuncia il nome dei menu: *calendario, programma, orario;* allora viene dapprima effettuato un controllo sull'effettiva selezione di un corso nella funzione *NessunCorsoSelezionato* (che ritorna un valore *True* se non è stato selezionato nessun corso), e successivamente viene chiamata una procedura che si preoccuperà di leggere e mostrare a video le informazioni corrispondenti. Se l'utente pronuncia il nome dei menu: risultato, prenotazione; allora viene effettuato un ulteriore controllo sull'effettivo inserimento di un numero di matricola nella funzione *NessunaMatricolaSelezionata* (che ritorna un valore *True* se non è stato inserito nessun numero di matricola).
Se l'utente pronuncia un numero, allora non si deve fare altro che mostrare a video il numero appena inserito (siamo nella fase di inserimento del numero di matricola).
Se l'utente pronuncia il comando fine, allora si legge un messaggio di ringraziamento e si ripuliscono i *TextBox* di selezione.
Se l'utente pronuncia un qualsiasi altro comando riconosciuto, allora si deve mostrare a video il corso appena selezionato (siamo nella fase di selezione del corso).
Lo schema delle procedure che si preoccupano di leggere e mostrare a video le informazioni selezionate è simile: si definisce una variabile che contiene il nome del file da leggere, si assegna il valore corrispondente a tale variabile, ed infine si chiama la procedura (*LeggiFile*) che dovrà effettivamente leggere e mostrare a video le informazioni selezionate. L'unica procedura differente è la procedura *EseguiMenuPrenotazioni* che dovrà, al contrario delle altre, scrivere su file il numero di matricola inserito.

```
Private Sub EseguiMenuCalendario()
Dim nomeFile As String
nomeFile = App.Path + "\" + TextCorso + "Cal.txt"
LeggiFile nomeFile, ""
End Sub
Private Sub EseguiMenuRisultati()
Dim nomeFile As String
nomeFile = App.Path + "\" + TextCorso + "Ris.txt"
LeggiFile nomeFile, TextMatricola
End Sub
Private Sub EseguiMenuPrenotazioni()
Dim nomeFile As String
nomeFile = App.Path + "\" + TextCorso + "Pre.txt"
ScriviSuFile nomeFile, TextMatricola
TextToSpeech1.Speak "Prenotazione Effettuata"
```

☑ PROPRIETÀ ED EVENTI DEL CONTROLLO VT

Il controllo VT espone tre proprietà:

- ***AnswerAfterRings*** **per fissare il numero di squilli prima di rispondere alla telefonata.**
- ***Initialized*** **per inizializzare il controllo.**
- ***MaxLines*** **per fissare il massimo numero di linee telefoniche.**

E genera quattro eventi:

- ***ClickIn*** **generato quando si clicca con il mouse sull'icona del controllo VT.**
- ***DoPhoneCall*** **generato alla ricezione della chiamata telefonica.**
- ***Initialize*** **generato quando nell'applicazione viene settata la proprietà initialized.**
- ***Shutdown*** **generato quando termina un thread associato ad una linea telefonica.**

Nell'evento *DoPhoneCall* dovrà essere contenuto il codice necessario a regolare l'intero colloquio telefonico con l'utente.

☑ COMPONENTI DI UN SISTEMA DI RV

Dispositivo per l'acquisizione della voce: è costituito da un microfono e da un convertitore A/D che codifica in forma digitale la forma d'onda vocale.

• *Digital Signal Processor (DSP):* l'obiettivo fondamentale di questo modulo è la produzione di una rappresentazione compressa della forma d'onda, estraendo quei componenti della rappresentazione spettrale che sono più significativi durante la fase di riconoscimento.

• *Pattern di riferimento:* contiene i modelli vocali di riferimento. Questi modelli si differenziano in base alle unità vocali utilizzate: fonemi, sillabe, parole, frasi.

• *Modulo di riconoscimento:* effettua il confronto tra le parole preprocessate pronunciate dal parlante e i modelli di riferimento preregistrati, selezionando tra le parole ammissibili, determinate dai vincoli grammaticali e lessicali (vocabolario), quella di massima similarità misurata in uno spazio metrico di riferimento.

```
End Sub
Private Sub EseguiMenuProgramma()
Dim nomeFile As String
nomeFile = App.Path + "\" + TextCorso + "Pro.txt"
LeggiFile nomeFile, ""
End Sub
Private Sub EseguiMenuOrario()
Dim nomeFile As String
nomeFile = App.Path + "\" + TextCorso + "Ora.txt"
LeggiFile nomeFile, ""
End Sub
```

La procedura *LeggiFile* si dovrà preoccupare di leggere le informazioni contenute nel file passato come parametro e, nel caso venga passato anche una matricola non vuota, dovrà leggere soltanto la riga di informazione corrispondente alla matricola selezionata.

```
Private Sub LeggiFile(nomeFile As String, matricola
                                          As String)
Dim StringaDaLeggere As String
Dim StringaTemp As String
Open nomeFile For Input As #1
Do While Not EOF(1)
  Input #1, StringaTemp
  If matricola = "" Then
     StringaDaLeggere = StringaDaLeggere
                                     + StringaTemp
  Else
     If Left(StringaTemp, 5) = matricola Then
        StringaDaLeggere = "la matricola numero "
              + matricola + " risulta " + Right(
           StringaTemp, Len(StringaTemp) - 5)
        Exit Do
     End If
  End If

Loop
Close #1
TextRisultato.Visible = True
TextRisultato = StringaDaLeggere
TextToSpeech1.Speak StringaDaLeggere
End Sub
Private Sub TextToSpeech1_SpeakingDone()
TextRisultato.Visible = False
End Sub
```

La procedura *ScriviFile* si dovrà preoccupare di scrivere sul file del corso selezionato, il numero di matricola inserito dallo studente:

```
Private Sub ScriviSuFile(nomeFile As String, matricola
                                          As String)
Open nomeFile For Append As #1
Write #1, matricola
Close #1
End Sub
```

Infine, per completezza, riportiamo le funzioni che controllano l'avvenuta selezione di un corso o di un numero di matricola

```
Private Function NessunCorsoSelezionato() As Boolean
If TextCorso = "" Then
   TextToSpeech1.Speak "Attenzione Nessun Corso
                                    Selezionato"
   NessunCorsoSelezionato = True
Else
   NessunCorsoSelezionato = False
End If
End Function
Private Function NessunaMatricolaSelezionata()
                                       As Boolean
If TextMatricola = "" Then
   TextToSpeech1.Speak "Attenzione Nessuna
                            Matricola Selezionata"
   NessunaMatricolaSelezionata = True
Else
   NessunaMatricolaSelezionata = False
End If
End Function
```

Naturalmente, quando l'applicazione viene chiusa, si deve rilasciare la memoria occupata dal processo di RV, perciò nella *Form_unload* scriviamo

```
Private Sub Form_Unload(Cancel As Integer)
  Vcommand1.ReleaseMenu CommandMenu
End Sub
```

Per l'uso dell'applicazione da parte di un qualsiasi utente non è necessario nessuna particolare fase di autoistruzione del parlante, poiché i comandi non sono molti e sono ben distinti tra essi, per questo si riduce la possibilità di errore.
Se la lista dei corsi aumenta o presenta corsi dal nome simile è necessario migliorare l'interazione con il motore di riconoscimento utilizzando i metodi: *GeneralDlg* per visualizzare la maschera di dialogo generale dello SE, oppure *TrainGeneralDlg* e *TrainMenuDlg* (che però non è disponibile per tutti gli SE) per istruire lo SE con caratteristiche particolari della voce, è infatti consigliata una breve fase di training se si hanno forti inflessioni dialettali.

CONCLUSIONI

L'applicazione realizzata nell'articolo è un punto di partenza che serve a mostrare come la tecnologia vocale sia alla portata dei programmatori VB.
A questo punto non resta che dare spazio alla fantasia per la realizzazione delle molteplici applicazioni possibili.

Ing. Luigi Buono

Creare oggetti C# da un documento XML

Importare documenti XML con C#

parte prima

Oggigiorno XML è lo standard di fatto per lo scambio di dati tra differenti sistemi e piattaforme. Per tale motivo, fornire un meccanismo per creare oggetti a partire da codice XML potrà risultare utile in numerosi contesti.

In questa serie di articoli dedicati a XML e C# utilizzeremo l'oggetto *XmlReader* e la reflection per realizzare un robusto e potente componente di utilità generale in grado di effettuare il mapping tra dati XML e oggetti C#. L'oggetto *XmlReader* rappresenta la modalità standard con cui un'applicazione .NET legge il contenuto di documenti XML. Mediante la *reflection* invece, le applicazioni possono ispezionare propria la struttura interna, che può essere usata per creare istanze a run-time, invocare metodi dinamicamente o reperire informazioni circa classi e relativi membri.

PERCHÉ IMPORTARE XML IN OGGETTI

Come abbiamo già detto, l'idea è quella di sviluppare una classe C# per importare codice XML scritto secondo un prefissato schema.
Questa classe, usando le informazioni contenute nel documento XML, sarà in grado (mediante la *reflection*) di creare dinamicamente istanze di classi che contengono proprio le informazioni presenti nel documento XML.
Consideriamo, per esempio, il seguente codice XML:

```
<class name="Persona">
  <field name="nome" value="Mario" type="string"/>
  <field name="cognome" value="Rossi" type="string"/>
</class>
```

La classe C# che importerà tale XML, creerà un'istanza della classe *Persona* in cui:

```
persona.nome = "Mario"
persona.cognome = "Rossi"
```

Ovviamente questo è un esempio molto banale. In situazioni più complesse il documento XML potrebbe contenere altre classi annidate o array d'oggetti. Il nostro componente dovrà essere in grado di funzionare anche in questi casi più complicati. Ad una prima occhiata, importare codice XML usando la reflection potrebbe non sembrare una strategia molto intelligente poiché il framework .NET mette a disposizione la serializzazione proprio per tale scopo. La serializzazione è una caratteristica molto potente in .NET e, a differenza di quella presente in Java, consente anche di serializzare e deserializzare oggetti in formato XML. Nonostante questo, come vedremo, esistono alcune situazioni nelle quali usare la serializzazione potrebbe non essere la strategia più appropriata. Per esempio, un caso potrebbe essere quando si ha la necessità di importare informazioni da sistemi esterni, che magari producono codice XML differente da quello interpretabile dalla serializzazione .NET. Per risolvere problemi come questo, abbiamo sostanzialmente due scelte: trasformare il codice XML in altro codice XML (usando XSLT) in formato compatibile con la serializzazione oppure creare un componente che legge il codice XML e crea oggetti C# in base alle informazioni in esso contenute. Il primo metodo può essere un'efficace e potente strategia, ma è in ogni caso necessario seguire la seconda strada se si è interessati anche al tipo dei dati. Infatti, la serializzazione .NET non consente di specificare il tipo di dato e questo a volte può rappresentare una significante limitazione, specialmente in sistemi in cui il mapping dei tipi è una problematica che non può essere tralasciata.

IL PROBLEMA

Lo scopo finale di questa serie di articoli su XML e

C# è quello mostrare come scambiare dati tra due sistemi, nei quali il sistema che importa è implementato in C# e quello che esporta fornisce informazioni usando codice XML che rispetta un predeterminato schema. Il sistema che legge il codice XML deve importare e creare i dati in accordo con il proprio design. Molto spesso, in sistemi orientati agli oggetti, importare significa proprio creare oggetti. Per realizzare ciò, si deve prevedere un meccanismo in grado di trasferire dati dal formato XML in oggetti. Spesso, questo è lavoro molto delicato da realizzare, poiché tipicamente ogni classe ha una diversa struttura, diversi attributi e metodi. Per esempio, la classe *Persona* avrà una struttura completamente differente dalla classe *Fattura*. La prima potrebbe avere attributi quali *nome, cognome* e *dataDiNascita*. La seconda invece possiederà attributi come *numero, data* e *importo*. Per questo motivo, usando un approccio semplicistico, sarà necessario implementare del codice ad hoc per importare ogni differente tipo classe. Naturalmente, noi vogliamo evitare proprio questa situazione. In definitiva, cosa vogliamo quindi? Ci piacerebbe creare una classe C# di uso generale in grado di leggere codice XML, interpretarlo e creare l'oggetto opportuno. Inoltre, tale classe dovrà quindi:

- **creare istanze di qualsiasi classe C#**
- **importare XML scritto con uno schema predefinito**
- **essere indipendente dalle classi da mappare (nessuna alterazione deve essere fatta al codice sorgente di quest'ultime)**
- **supportare i tipi di dato**

Questo è proprio quello che realizzeremo in questa serie di articoli.

PROGETTAZIONE E SVILUPPO DEL COMPONENTE

Adesso che il problema è stato definito, possiamo iniziare a svilupparne la soluzione. Prima di tutto sarà necessario introdurre alcune caratteristiche che il framework .NET mette a disposizione e che ci saranno di grandissima utilità. Per prima cosa, dobbiamo capire come C# e .NET dialogano con XML; secondo, bisogna dare un'occhiata al modo in cui, mediante la reflection, è possibile creare dinamicamente delle istanze di classi. Come è già stato detto, l'idea base è quella di sviluppare un componente che legge codice XML che rispetta un particolare schema. Nello stesso momento in cui il componente legge i vari elementi (*parsing*), esso deve essere in grado di creare l'opportuno oggetto C# e riempirne i campi con i dati provenienti dal documento XML. Il nome della classe, quello degli attributi ed i valori che assumono, sono scritti all'interno dell'XML, quindi il componente avrà tutti i dati necessari per creare, usando la reflection, le istanze delle varie classi e ritornarne un riferimento.

XML PARSER E XMLREADER

La classe *XmlReader* è una classe astratta che fornisce un meccanismo ad alte prestazioni (non dotato di cache) per dialogare con codice XML. La classe *XmlTextReader* è un'implementazione concreta di *XmlReader* che consente di accedere e leggere codice XML presente in uno stream di testo. L'utilizzo di *XmlTextReader* è la modalità standard per leggere il contenuto di un documento XML. Questa classe non aderisce né alla filosofia SAX né a quella DOM. Comunque, il modo di operare di *XmlTextReader* è molto più vicino a SAX che a DOM, poiché la classe effettua il parsing del codice XML consentendo alle applicazioni di processare gli elementi nello stesso momento in cui questi vengono letti. A differenza di DOM, un oggetto *XmlTextReader* non crea nessun modello ad oggetti in memoria per rappresentare la struttura del codice XML. Il modo in cui leggere un documento XML, mediante *XmlTextReader*, è molto simile al modo in cui si legge il contenuto di un database con un cursore. Tale cursore si sposta sui vari elementi; l'applicazione potrà quindi di volta in volta, fare uso del contenuto dell'elemento e dei suoi attributi. Generalmente il contenuto di un documento XML può essere visto come una rappresentazione ad albero. Il cursore visiterà l'albero adottando la tecnica *in-depth first:* partendo dalla radice di un qualunque sottoalbero il nodo successivo da visitare sarà il primo a sinistra. Appena la visita arriva ad una foglia, si risale al padre per visitare un eventuale "fratello". Se non sono presenti nodi fratelli, la visita risale ancora e procede nello stesso modo. Il modello a cursore, utilizzato dalla piattaforma .NET per leggere il contenuto di un file XML, è di tipo *read-only forward-only.* Non sono quindi ammesse scritture e la visita può procedere solo in avanti (non sarà possibile ritornare indietro per visitare nodi già visitati). Usando la classe *XmlTextReader*, è possibile sviluppare rapidamente un parser XML che verifica se un documento è well-formed, ovvero se rispetta la sintassi imposta da XML. Se vogliamo eseguire anche un controllo semantico del documento, si deve utilizzare invece la classe *XmlValidatingReader*. Essa è una specializzazione di *XmlReader* che consente di specificare il tipo di validazione che si intende utilizzare. Alcune delle possibili scelte sono: *nessuna, DTD* o *XML schema (XSD)*. La prima operazione, per

☑ SYSTEM.XML

Per interoperare con XML, .NET mette a disposizione il namespace *System.Xml* che contiene un insieme di classi ed interfacce il cui scopo principale è quello di leggere, manipolare e scrivere codice XML. Le classi che utilizzeremo in quest'articolo saranno *System.Xml.XmlReader* ed alcune delle sue derivate come *XmlTextReader* e *XmlValidatingReader*.

implementare un parser XML validante, è creare gli oggetti *XmlTextReader* e *XmlValidatingReader*:

```
XmlTextReader txtReader = new XmlTextReader(filename);
XmlValidatingReader xtReader =
                    new XmlValidatingReader(txtReader);
```

Come si può vedere, abbiamo bisogno di entrambe le istanze *(XmlTextReader* e *XmlValidatingReader),* perché *XmlValidatingReader* non estende direttamente *XmlTextReader.* Per istanziare *XmlValidatingReader* è necessario passare un'istanza di *XmlTextReader* come parametro del costruttore. È possibile impostare il tipo di validazione grazie alla proprietà *ValidationType*, come segue:

```
xtReader.ValidationType = ValidationType.Schema;
```

La proprietà *ValidationType* è un'enumerazione. I valori previsti sono: *Auto, DTD, None, Schema* e *XDR*. Noi utilizzeremo sempre e solo la validazione di tipo *Schema* (che fa uso di schemi XML). Il passo successivo è quello di definire il gestore dgli eventi (*event handler*) per la validazione, che sostanzialmente è un metodo richiamato dal framework ogni qualvolta che un errore di validazione si verifica. Il modo per impostare l'handler per la validazione è il seguente:

☑ XML TEXTREADER

Usando la classe *XmlTextReader*, è possibile sviluppare rapidamente un parser XML che verifica se un documento è well-formed, ovvero se rispetta la sintassi imposta da XML.

```
xtReader.ValidationEventHandler +=
 new ValidationEventHandler (this.ValidationEventHandle);
```

La funzione invocata sarà quindi *ValidationEventHandle.* Tra breve, ne vedremo l'implementazione. A questo punto sarà possibile invocare il metodo *Read* della classe *XmlValidatingReader* che inizierà a leggere il documento. Il metodo cesserà la propria esecuzione ogni qualvolta un'*entità* XML viene letta. Per entità si intendono elementi, attributi o tag XML. L'istanza *xtReader* conterrà, di volta in volta, tutte le informazioni relative all'ultima entità letta. Il metodo restituirà *false* se non ci sono ulteriori entità da leggere. In questo primo esempio invocheremo il metodo *Read* fin quando non restituirà *false*, quindi le informazioni presenti in *xtReader* non saranno usate:

```
while (xtReader.Read()) { }
xtReader.Close();
```

L'implementazione dell'handler per la validazione è la seguente:

```
public void ValidationEventHandle (object sender,
                          ValidationEventArgs args)
{ throw new System.Xml.XmlException(args.Message,null);}
```

Quando avviene un errore, l'handler per la validazione lancerà un'eccezione il cui messaggio sarà prelevato dal parametro *args*.

L'XML IMPORTER

Mettendo insieme i frammenti di codice C# visti in precedenza, si può realizzare una classe che funzionerà da parser XML validante. Questo però non sarebbe sufficiente ai nostri scopi. Infatti, è nostra intenzione implementare una classe base che potrà poi essere utilizzata in seguito nel corso di questa serie di articoli. Essa dovrà essere in grado di effettuare il parsing di un documento XML e delegare ad un'altra classe, definita dall'utente, la possibilità di processare le entità XML che saranno lette di volta in volta. Quello che andremo a sviluppare rispetterà il class diagram presente in Fig. 1. La figura mostra la classe *XmlImporter*, che è il nostro obiettivo finale, e le relazioni che essa ha con altre classi ed interfacce. Sostanzialmente, l'*XmlImporter* è un parser XML validante basato sugli schemi XML che legge (mediante il metodo *import*) il documento XML ed invoca un metodo definito dall'utente ogni volta che un elemento deve essere processato. Questo metodo, chiamato *processElement,* appartiene all'interfaccia *Handler.* Come si può vedere dal *class diagram, XmlImporter* possiede un oggetto *Handler.* L'interfaccia *Handler* sarà così definita:

```
public interface Handler {
// processElement è chiamata dell'XmlImporter
// ogni volta che un elemento deve essere processato
void processElement(XmlValidatingReader xtReader);
// processObject sarà invocato dall'utente
// per processare l'oggetto importato
void processObject(object obj);}
```

Il metodo *processElement*, come già menzionato, è richiamato dall'*XmlImporter* ogni volta che un elemento XML viene letto. Il parametro *xtReader*, rappresenta un *XmlValidatingReader* che contiene tutte le informazioni dell'ultimo elemento XML letto. Di conseguenza, sarà possibile implementare processElement in modo da utilizzare i dati provenienti dal documento XML nel modo che si ritiene opportuno. Il metodo *processObject* sarà invocato appena un oggetto C# viene creato a partire dal codice con-

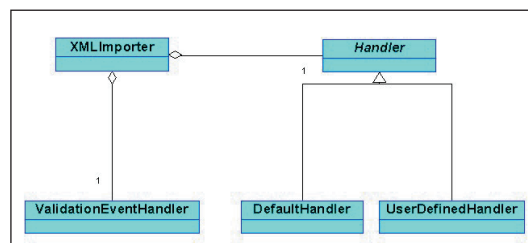



***Fig. 1:* Class diagram relativo alla classe XmlImporter.**

tenuto nel documento XML. Implementando questo metodo, l'applicazione potrà fare uso dell'oggetto per i propri scopi. Per utilizzare l'*XmlImporter*, è obbligatorio implementare l'interfaccia *Handler*. È prevista, in ogni modo, un'implementazione di default chiamata *DefaultHandler* (il codice è presente sul CD o sul Web) che si comporta semplicemente come un parser XML validante. Dal diagramma si può notare che *XmlImporter* è aggregata con *ValidationEventHandler* che, nonostante il fatto sia modellato come una classe, in realtà è un metodo, lo stesso che abbiamo visto nella sezione precedente.
A questo punto possiamo implementare *XmlImporter*. Ecco il codice:

```
using System;
using System.Xml;
using System.Xml.Schema;
namespace Wrox.Xml.Mapping {
  public class XmlImporter {
    // Handler definito dall'utente
    private Handler handler;
    // Construttore (usa il default handler)
    public XmlImporter() {
      this.handler = new DefaultHandler(); }
    // Construttore (usa l'handler in input)
    public XmlImporter(Handler handler) {
      this.handler = handler; }
    // Legge il file XML
    public void import(string strFilename) {
      // XmlReader
      XmlTextReader txtReader =
        new XmlTextReader(strFilename);
      XmlValidatingReader xtReader =
        new XmlValidatingReader(txtReader);
      // Imposta la ValidationType a XML-Schema (XSD)
      xtReader.ValidationType = ValidationType.Schema;
      // Imposta il validation handler
      xtReader.ValidationEventHandler +=
        new ValidationEventHandler
         (this.ValidationEventHandle);
      try {
        // Parsing
        while (xtReader.Read()) {
          // Invoca processElement
          handler.processElement(xtReader); } }
      catch (Exception e) {
       throw new Exception(e.Message);}
      finally { xtReader.Close();} }
    // Validation Event Handle
    public void ValidationEventHandle (object sender,
       ValidationEventArgs args) {
      throw new System.Xml.XmlException
       (args.Message,null); } }}
```

La classe presenta due costruttori: il primo crea un'istanza di *XmlImporter* usando il *DefaultHandler*, il secondo, invece, consentirà di passare l'*Handler* come parametro. Il metodo *import* prende in input il nome del file XML, ne effettua il parsing - utilizzando l'*XmlValidatingReader* e, per ogni elemento XML, invoca il metodo *processElement* dell'Handler. Di seguito, è riportata usa semplice implementazione di *Handler* (chiamato *MereHandler*, ovvero Handler semplice) che visualizza il nome degli elementi XML e dei relativi attributi:

```
public class MereHandler : Handler {
  public void processElement(XmlValidatingReader
    xtReader){
    if (xtReader.NodeType==XmlNodeType.Element){
      Console.WriteLine("Element: " + xtReader.Name);
      for (int i=0;i<xtReader.AttributeCount;i++) {
        xtReader.MoveToAttribute(i);
        Console.WriteLine("    attribute {0} = {1}",
          xtReader.Name,xtReader.Value); } }
  public void processObject(object obj){} }
```

Come si può notare, il primo controllo è che l'oggetto *xtReader* contenga informazioni relative ad elementi, in altre parole che la proprietà *NodeType* sia uguale a *XmlNodeType.Element*. L'enumeration *XmlNodeType* indica il tipo di entità XML: *Element, Attribute, Comment, Document* e così via. Dopo tale controllo, il nome dell'elemento e i suoi attributi saranno visualizzati sulla console. Il seguente frammento di codice istanzia un oggetto *XmlImporter* (usando *MereHandler*) ed invoca il metodo *import* passando il file chiamato *example.xml*:

```
xmlImporter = new XmlImporter(new MereHandler());
xmlImporter.import("example.xml");
```

L'applicazione visualizzerà gli elementi e i relativi attributi contenuti nel file. Dovrebbe essere chiaro a questo punto che, implementando opportunamente l'interfaccia *Header*, ed in particolare il metodo *processElement*, possiamo trasferire codice XML in oggetti C#. Infatti, mediante l'oggetto *xtReader* abbiamo a disposizione tutte le informazioni che ci servono: il nome della classe, i campi ed i relativi valori. Come possiamo però creare tali oggetti dinamicamente partendo dal nome e valorizzarli con le corrette informazioni? La risposta a questa domanda va ricercata all'interno di una *feature* molto potente: la *reflection*.

CONCLUSIONI

In questo primo articolo dedicato a XML e C# abbiamo visto come leggere il contenuto di un documento XML usando l'*XmlReader*. Nel prossimo articolo vedremo come usare il parser XML e la reflection al fine di importare oggetti semplici.

Giuseppe Naccarato

☑ BIBLIOGRAFIA

• **MAPPING XML TO C# OBJECTS USING REFLECTION**
G. Naccarato
C#Today
(Wrox Press)
Ottobre 2002

• **PROFESSIONAL C# 2ND EDITION**
S. Robinson e altri
Wrox Press 2002

• **.NET SERIALIZATION**
S. Gopikrishna,
K. Sanghavi
C#Today
Luglio 2001
http://www.csharptoday.com/content.asp?id=1532

• **REFLECTION IN .NET**
Hari Shankar
C# Corner
Febbraio 2001
http://www.c-sharpcorner.com/1/Reflection_in_net.asp

• **C# PROGRAMMING: ATTRIBUTES AND REFLECTION**
Jesse Liberty
(XML.com)
Agosto 2001
http://www.xml.com/pub/r/1207

Tecnologie: l'integrazione tra Flash MX e Web Service

Annunci on-line con Web Service e Flash

L'unione della potenza di Flash MX con la flessibilità offerta dai Web Service costituisce un'occasione imperdibile per qualsiasi sviluppatore.

Stiamo lentamente assistendo alla sostituzione di vecchie tecnologie e vecchi modi di concepire il web. Lato client, a non funzionare sono le interfacce utente scomode, difficilmente manutenibili e non sempre compatibili con i vari user agent. Lato server, troppo spesso ci troviamo di fronte ad architetture poco flessibili e quasi mai scalabili nella realtà quotidiana. Intere applicazioni funzionano solo con se stesse rendendo poco agevole la coesistenza con altri prodotti se non vietandone di fatto l'introduzione. Vengono adottati pochi standard, quasi sempre a causa dei brevi tempi di sviluppo a disposizione per le prime consegne. Ancora lato client, l'HTML sembra non morire mai, anzi, è sempre molto vivo.
Con DHTML si possano creare effetti carini ma non sempre è facile creare una perfetta ergonomia per le proprie applicazioni web. Troppi limiti: compatibilità, molti giorni di sviluppo buttati per scoprire che un browser visualizza i dati disallineati, etc. Schematicamente, possiamo riassumere che Flash Mx risolve i problemi di ergonomia e di compatibilità, mentre i Webservices quelli di poca compatibilità e di applicazioni poco scalabili. I WebServices sono dei componenti software che permettono, tramite protocolli standard e aperti (ad es. SOAP) basati sull'utilizzo di XML, di creare delle applicazioni che utilizzino la porta 80 per veicolare dati e funzioni. Possono essere invocati da chiunque e sono indipendenti dalla piattaforma. Tramite essi, sistemi completamente diversi possono comunicare fra loro. Paradossalmente sistemi in cobol potrebbero dialogare con php. Grazie a questi principi, complesse architetture possono essere suddivise in piccoli oggetti modulari, permettendo di creare framework come se si stesse disegnando un sistema oop con evidenti vantaggi in termini di eleganza, scalabilità e scarsità di bachi. Essi però non vanno usati come il prezzemolo, ma solamente dove le policy di sicurezza lo consentano, dove non servono grosse prestazioni, e dove ha senso usarli ovvero quando si devono far interagire sistemi diversi scritti in linguaggi di programmazione diversi. Inoltre, c'è un altro importante fattore sempre più attuale: la multicanalità. Trovandoci di fronte a numerosi device, ognuno con display diverso, come sviluppatori siamo obbligati a scindere i dati dalla logica di presentazione. In un sito standard una soluzione sarebbe quella di usare xml + xsl, noi vedremo come farlo in Flash, visto che già esso funziona con successo su device di nuova generazione. Costruiremo un esempio che ci permetterà di raccogliere e visualizzare degli annunci tramite WebServices. Costruiremo il WebService con Coldfusion, perché magicamente riusciremo a condensare tutta l'applicazione lato server in una decina di righe di codice e qualche settaggio. Per dimostrare di aver scritto qualcosa di veramente standard, tramite php invocheremo il servizio di Coldfusion. L'interfaccia sarà costituita da un movie Flash costruito con delle classi Actionscript in grado di separare nettamente i dati da come vengono presentati. Flash è vettoriale e si adatta bene a diverse dimensioni e questo sarà un vantaggio aggiuntivo. Le volte precedenti abbiamo già visto come Flash Mx può invocare un webservice ora non ci resta che ripercorrere pochi semplici passi per crearcene uno da zero.

PREPARAZIONE AMBIENTE

1) Database

Come database useremo Access. Creiamo una nuova tabella con questi campi: *Codice, Categoria, Cap, Testo, Inserzionista, Telefono, Email, Prezzo.* Per semplicità non useremo nessuna procedura di normalizzazione e velocemente settiamo il campo Codice

☑ SCAMBIO DATI

Un'applicazione web basata su webservices prevede uno scambio di stream xml formattati secondo lo standard soap. Per far funzionare un webservice fondamentalmente serve solo la capacità di generare richieste HTTP POST a un web server.

Nome campo	Tipo dati
Codice	Contatore
Categoria	Testo
Cap	Numerico
Testo	Testo
Inserzionista	Testo
Telefono	Testo
Email	Testo
Note	Testo

Fig. 1: ***Access ha dalla sua una estrema semplicità e una capillare diffusione.***

come contatore, cap e prezzo come numerici e il resto dei campi come testo. Salviamo la tabella con il nome tbl_annunci. Chiudiamo il database salvandolo con nome *"db.mdb"*.

2) Configurare Coldfusion

Andremo poi ad installare il motore di tutto ovvero ColdFusion Mx (scaricabile dal sito macromedia in prova) che è di immediata installazione, non richiede nessuna configurazione, basta solo lanciare il setup, leggere le informazioni che si presentano e scegliere avanti, avanti… fine. Ricordatevi le password che scegliete in fase di installazione perché quando consulterete l'amministrazione vi verranno chieste. Per collegarci al pannello di amministrazione di Coldfusion Mx nel nostro browser carichiamo questa pagina: *http://localhost:8500/CFIDE/administrator/index.cfm*. (localhost o l'indirizzo della macchina in cui è avvenuta l'installazione). Se la pagina si carica significa che tutto è andato a buon fine. Ora abbiamo l'ambiente pronto; non ci resta che informare Coldfusion Mx della presenza del database.

☑ CONDIVIDERE

Aziende che già usano componenti java o com utilizzando Coldfusion possono costruirsi webservices in modo molto rapido: Coldfusion è disponibile anche per piattaforme J2EE. In questo modo i propri servizi possono essere condivisi facilmente senza lavoro aggiuntivo.

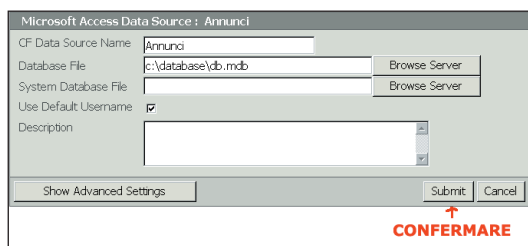



Fig. 2: ***Colleghiamo il DB a Coldfusion.***

Per questo, dall'amministrazione clicchiamo sul menu *Data Sources* specificando nella maschera foto che compare come nome *Annunci*, come driver *Access* e scegliendo il nostro file *db.mdb*. D'ora in poi faremo riferimento a questa sorgente dati.

3) Componente lato server

Scriviamo ora il nostro componente (CFC). Tramite dei semplici tag di alto livello esso implementa questi metodi:

- ***InserisciNuovo:*** **inserisce un nuovo annuncio;**
- ***SelezionaTutti*****: si occupa di selezionare la lista di annunci disponibili;**
- ***DammiDettaglio*****: estrapola i dettagli di un annuncio.**

All'interno del nostro componente per ogni metodo lo schema è: definizione funzione server con tipo di ritorno, definizione query con relativo datasource. Ecco un esempio:

```
<cffunction name="selezionaTutti" access="remote"
                                    returnType="query">
 <CFQUERY NAME="q_annunci"
                              Datasource="Annunci">
   SELECT Codice, Categoria, Cap, Testo, Inserzionista,
                                 Telefono, Email, Prezzo
   FROM tbl_annunci
 </CFQUERY>
 <cfreturn q_annunci>
</cffunction>
```

In questo modo dichiariamo il metodo *selezionaTutti* (che poi invocheremo dal movie flash), specificando la query da eseguire sul db. Il recordset risultante dall'operazione di selezione sarà il ritorno della funzione. A mio avviso, qui notiamo subito una cosa fondamentale: tutto il codice lato server si condensa in pochissime righe in quanto nel componente non è racchiusa alcuna logica di presentazione e visualizzazione tipica delle pagine html.

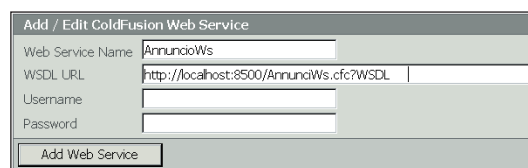



Fig. 3: ***Creare WebServices con Coldfusion è facile come un clic!***

Il codice Coldfusion infatti è semplice e conciso in quanto focalizza l'attenzione soltanto alla comunicazione "Client Flash" – "Server" direttamente tramite recordset. Gli altri metodi seguono la stessa struttura. Allegato alla rivista troverete l'esempio appena analizzato e il componente intero. Salvate il file *"Annunci.cfc"* nella vostra *wwwroot* (se avete lasciato tutti i default essa corrisponde a *[directory_installazione_coldfusion] \wwwroot*). Digitate nel vostro browser *http://localhost:8500/Annunci.cfc*, Coldfusion genererà una scheda di riepilogo (ottima per la documentazione) con la lista dei metodi e relativi parametri da noi implementati.

4) Trasformazione componente in webservice
A questo punto dobbiamo trasformare il nostro componente in webservice e anche quest'operazione grazie a Coldfusion avverrà gratis. Infatti basta scegliere la voce "Web Services" dal menu dell'amministrazione e specificare nel pannellino come nome *"AnnuncioWs"* e come WSDL URL. *http://localhost:8500/Annunci.cfc? WSDL.* Coldusion genererà per voi persino il WSDL! Per vederlo caricate nel browser la WSDL-url appena aggiunta.

dammiDettaglio

```
remote query dammiDettaglio ( Codice )

Output: enabled
Parameters:
  Codice: any, optional, Codice
```

inserisciNuovo

```
remote string inserisciNuovo ( Categor

Output: enabled
Parameters:
  Categoria: any, optional, Categoria
  Cap: any, optional, Cap
  Testo: any, optional, Testo
  Inserzionista: any, optional, Inserzionista
  Telefono: any, optional, Telefono
  Email: any, optional, Email
  Prezzo: any, optional, Prezzo
```

selezionaTutti

```
remote query selezionaTutti ( )

Output: enabled
```

Fig. 4: ***Il dettaglio dei metodi appena creati.***

MOVIE FLASH

Il nostro movie deve essere "multicanale" e per questo deve essere in grado di comportarsi in modo diverso a seconda del contesto. Tutte le classi devono essere costruite per separare nettamente il dato da come è rappresentato seguendo una logica OOP ordinata. Noi ci serviremo di due classi actionscript principali:

WebServiceClass
comunica col server invocando il webservice per estrapolare i dati dal db. Si occupa di connettersi al gateway tramite l'oggetto *NetConnection* e di creare un riferimento al web service utilizzando il metodo *getService().*

PresenterClass
si occupa di disegnare l'interfaccia grafica dei dati prelevati da *WebServiceClass*

Evitando di usare oggetti non ancora del tutto documentati di Flash Mx per far comunicare le due classi ho scritto una superclasse dalla quale ereditano entrambe e che ci fornisce un efficace sistema di messaggistica basata su eventi. Vediamo come:

```
// Definisco classe gestione eventi inizializzando un
                                     vettore di ascoltatori
EventManager = function () {
  this.listeners= new Array ();
}
// metodo di aggiunta di un ascoltatore
EventManager.prototype.addListener = function (obj) {
  this.listeners.push (obj);
}
// metodo che esegue l'evento -inteso come metodo-
                                 a tutti i nostri ascoltatori
EventManager.prototype.sendEvent = function (event,
                                                  param) {
  for (var i in this.listeners)
    this.listeners[i][event](param);
}
```

L'istruzione *this.listeners[i][event](param)* esegue su tutti i propri ascoltatori il metodo specificato dall'evento passando il parametro necessario. L'ereditarietà la settiamo in actionscript in questo modo:

```
WebServiceClass.prototype= new EventManager ();
PresenterClass.prototype= new EventManager ();
```

Così facendo le due classi acquisiscono i metodi *addListener* e *sendEvent*, vediamo come vanno usati:

tramite il metodo addListener registriamo le istanze delle due classi vicendevolmente.

tramite il metodo sendEvent facciamo "transitare" il messaggio. Supponiamo di aver ricevuto la lista di annunci tramite la *WebserviceClass*; usando il remoting la ricezione avviene implementando questo metodo (notare la sintassi *nomemetodoserver_result*):

```
// callback di risposta metodo di selezione annunci
WebServiceClass.prototype.selezionaTutti_Result=
                                        function (result) {
        // informo l'oggetto grafico che mi rappre-
senta di disegnare la lista con i dati ricevuti
        this.sendEvent ("Draw", result);
}
```

Qui viene il bello: il metodo sendEvent eseguirà il metodo *Draw* sul suo *Presenter* dicendogli con che recordset disegnarsi.

☑ WSDL: GENERAZIONE AUTOMATICA
Quando si trasforma un componente Coldfusion in webservice bisogna verificare che l'attributo access del tag cffunction sia settato a "remote": grazie a questo Coldfusion saprà anche generare il file WSDL in automatico.

```
// metodo che disegna la lista di record (rs è il recordset)
PresenterClass.prototype.Draw = function (rs) {
  // ... serie di attachMovie di oggetti grafici in libreria
}
```

Il metodo *Draw* contiene una serie di *attachMovie* che inseriscono nello stage una serie di oggetti grafici presenti in libreria. Nell'esempio ho costruito un oggetto *record* che rappresenta un oggetto della lista: esso visualizza alcuni dati riassuntivi relativi all'annuncio ovvero inserzionista, categoria, cap, prezzo. Cliccando su ogni item il *Presenter* chiede alla *WebserviceClass* di interrogare il server e di dargli le informazioni aggiuntive tipo il testo esteso, l'email, il numero di telefono per contattare l'inserzionista etc. Analogamente al sistema di selezione quando clicchiamo sul pulsante inserisci tramite *WebServiceClass* invochiamo sul nostro webservice il metodo *inserisciNuovo(parametri)* passando i valori inseriti nel modulo. La gestione della presentazione grafica è lasciata tutta alla classe *PresenterClass*.

Inserzionista | Testo annuncio
e-mail
telefono
CAP | Prezzo
Categoria | INSERISCI

Inserzionista	Categoria	CAP	Prezzo
Demis Magoga	Immobili	30110	300000
Demis Magoga	Pc Hardwar	30110	100
Antonio Rossi	Pc Hardwar	32311	300
Massimo Verdi	Automobili	33111	20000
Massimo Verdi	Automobili	33111	20000
Giuseppe Scaturli	Barche	30100	3000

Fig. 5: ***La maschera di inserimento dei record con in basso la lista di quelli inseriti.***

Basterà modificare qualche riga sul metodo di disegno per stravolgere l'effetto sull'interfaccia grafica senza intervenire sulla parte che interagisce col server: questa caratteristica ci apre le porte allo sviluppo di movie multicanali. In questo modo infatti per creare interfacce grafiche completamente diverse basta cambiare soltanto gli oggetti in libreria e qualche costante!!! Vi sono altri parametri da considerare tipo la grandezza dello stage ma se siamo furbi possiamo sfruttare la "vettorialità" di flash adattando *width* ed *height* dei parametri *object/ embed* che si occupano di includere il flash in pagina. Lascio a voi il compito di estendere il movie base fornitovi come esempio. L'interfaccia scarna potrebbe essere arricchita con nuovi elementi insieme ai controlli sui tipi di dato in input; potrebbe essere aggiunto qualche sistema di cache lato client. Allegato alla rivista avete tutti i sorgenti commentati dai quali partire.
Divertitevi!

☑ NUSOAP

La creazione di un end-point soap è semplice con la libreria *nusoap*. Basta definire una serie di funzioni e registrarle presso un oggetto soap-server mentre per richiamarle basta definire un oggetto soap-client. La libreria open source scritta in php la si può trovare al seguente indirizzo:

http://dietrich.ganx4.com/nusoap/index.php

TEST DEL WEB SERVICE

Scopo del nostro webservice e di tutto il nostro lavoro è stato quello di costruire un servizio capace di dialogare con chiunque. Verifichiamo se è vero chiamando il nostro servizio utilizzando una tecnologia diversa, ovvero php, realizzando qualcosa di universale oltrechè multicanale. Per richiamare il webservice appena scritto con coldfusion o qualsiasi webservice da php si utilizza la famosa libreria nusoap. Il codice è il seguente:

```
<?
 require_once('nusoap.php');
 $soapclient = new soapclient('http://localhost:8500/
                                        Annunci.cfc?WSDL');
 $ret= $soapclient->call('selezionaTutti');
 ....
 for ($j=0; $j < count ($annunci[$i]); $j++) {
   // per ogni elemento record
   echo "<tr>";
    for ($k=0; $k < count ($annunci[$i][$j]); $k++)
      // per ogni campo
      echo "<td>".$annunci[$i][$j][$k]."</td>";
   echo "</tr>";
 }
  ....
?>
```

Dopo aver incluso la libreria nusoap creiamo un riferimento al nostro webservice tramite questa istruzione:

```
$soapclient = new soapclient('http://localhost:8500/
                                        Annunci.cfc?WSDL');
```

In seguito invochiamo il metodo di selezione annunci specificando il metodo implementato su Coldfusion:

```
$ret= $soapclient->call('selezionaTutti');
```

La variabile php $ret rappresenta il recordset. A noi non resta che scorrerlo e visualizzare una tabella. Abbiamo così fatto comunicare due diverse piattaforme: PHP e Coldfusion.
Grazie Webservice!

Demis Magoga

Elementi per la manipolazione delle stringhe

Regular Expressions con .NET

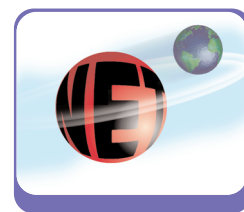
Le Regular Expressions sono strumenti potenti che possono apparire alquanto enigmatici. In questo articolo vengono esposti i concetti base per la loro comprensione e l'utilizzo nelle applicazioni .NET.

Le *Regular Expressions* (espressioni regolari) sono uno strumento molto potente e vengono utilizzate nei più svariati contesti. Nell'ambito dello sviluppo si sente parlare spesso di *regex* (usuale contrazione di Regular Expressions) suscitando odio e amore nei programmatori. Sono molto apprezzate per le loro potenzialità e per quel che permettono di realizzare, ma scriverle e comprenderle non è così immediato.
Si prenda in considerazione la seguente regex:

```
^((4\d{3})|(5[1-5]\d{2})|(6011))-?\d{4}-?\d{4}-?\d{4}|3 [4,7]\d{13}$
```

Come si può intuire facilmente, non deve esser stato così immediato generare una stringa di questo tipo. Al di là delle abilità individuali bisogna riconoscere che si tratta di una stringa poco leggibile e alquanto criptica e che anche la scrittura sensata di un tale *pattern* (modello) possa risultare un'operazione piuttosto ostica; eppure quella stringa, apparentemente senza senso, permette di verificare se il numero di carta di credito inserito da un utente sia, o meno, nel formato corretto delle carte di credito Visa, MasterCard, ecc…ecc… Lo stesso Robert Howard, (Program Manager di ASP.NET) sostiene di apprezzare le potenzialità delle regex, ma odia doverne scrivere di proprio pugno. In questo articolo verrà mostrato cosa sono le regex e come vengono utilizzate da applicazioni .NET, cercando di svelare il significato di tali "misteriose" sequenze di caratteri.

REGEX OVUNQUE

Le regex permettono la ricerca e la sostituzione di stringhe all'interno di altre stringhe utilizzando un pattern. La criptica stringa citata all'inizio è un pattern per le espressioni da confrontare mediante un'operazione detta di *match*. La maggior parte degli utenti con un certo background informatico, hanno utilizzato le regex senza neppure saperlo. Ad esempio se volessimo ricercare tutti i file eseguibili contenuti in una directory, potremmo utilizzare il pattern **.exe* nel file search di Windows. L'applicazione più diffusa delle regex è infatti la ricerca e sostituzione di stringhe in word processor o la ricerca mediante filtri in utilità di tipo "trova file". I campi di applicazione sono comunque molto più vasti e sempre correlati alla manipolazione di testo: vanno da programmi anti-spam che filtrano il traffico e-mail indesiderato, al controllo della correttezza del formato di inserimento dei dati nei form, al parsing di pagine web e file XML. Le regex sono insomma uno strumento flessibile, potente e molto efficiente per compiere operazioni su testi, anche molto estesi. Probabilmente il lettore ha familiarità con i meta-caratteri ***** e **?** usati per i comandi DOS o per le ricerche di file nel file-system di Windows, ma le regular expressions possono essere considerate un vero e proprio linguaggio basato su caratteri testuali e meta-caratteri. Le regex si basano sui meta-caratteri e sono organizzati in varie categorie: caratteri di escape, di sostituzione, classi di caratteri, quantifiers, costrutti di gruppo, di back-reference, di alternanza, ecc…ecc… A breve saranno esposti i principali meta-caratteri utili nell'uso quotidiano delle regular expressions.

META-CARATTERI

I caratteri . *$ {[^ ( ) | * + ?* \ sono dei caratteri speciali che hanno un particolare significato nelle regular expressions, tutti gli altri caratteri sono detti ordina-

ri e corrispondono a se stessi. Segue una tabella dei *caratteri di escape*, ovvero quelle combinazioni di caratteri che assumono significati di vario tipo nel pattern delle regex:

CAR.	SIGNIFICATO
\	Anteposto ad un carattere speciale lo rende ordinario (o *literal*), privandolo del suo significato particolare. Ad esempio \^ corrisponde esattamente al carattere ^.
\b	Carattere di backspace
\f	Avanzamento form (*form feed*)
\n	Nuova riga (*line feed*)
\r	Invio a capo (*carriage return*)
\t	Carattere di tabulazione
\v	Tabulazione verticale

Oltre a singoli caratteri, è possibile specificare una *classe di caratteri*, ovvero una serie di caratteri espressi da un costrutto. Segue una tabella riassuntiva dei principali costrutti:

CLASSE	SIGNIFICATO
.	Uno qualsiasi carattere, eccetto *\n*
[aeiou]	Un carattere tra quelli indicati
[^aeiou]	Un qualsiasi carattere tranne quelli indicati
\w	Carattere alfanumerico
\W	Carattere non alfanumerico
\s	Indica qualsiasi carattere di white-space, equivale a *[\f\n\r\t\v]*.
\S	Indica qualsiasi carattere non di white-space, equivale a *[^\f\n\r\t\v]*.
\d	Ogni carattere decimale
\D	Ogni carattere non decimale
[2-7c-nL-T]	Un carattere qualsiasi nel range di caratteri contigui in *2-7*, *c-n* o *L-T*.

☑ SOFTWARE

Un programma alquanto utile per la realizzazione e la verifica delle Regex è Regex Designer:

http://www.sellsbrothers.com/tools/

Concludendo questa breve carrellata dei metacaratteri principali, segue una tabella dei caratteri speciali (non ordinari, alcuni di essi sono *quantificatori*):

CAR.	SIGNIFICATO
$	Fine di una stringa
^	Inizio di un stringa
*	0 o più occorrenze dell'espressione precedente. Ad esempio *\w** indica 0 o più caratteri alfanumerici
?	0 o 1 carattere della precedente espressione
[...]	Set di caratteri
(...)	Gruppo di caratteri
{N}	Indica *N* occorrenze dell'espressione precedente
{N,}	Indica almeno *N* occorrenze dell'espressione precedente
{N,M}	Indica almeno *N* e al massimo *M* occorrenze dell'espressione precedente

UN PRIMO APPROCCIO

Se si costituisce un pattern con soli caratteri ordinari, la regular expression risultante permetterà la ricerca di ogni occorrenza della stringa stessa nel testo. Ad esempio se si volesse ricercare la stringa *"Cangiano"*, la regular expression sarebbe la stringa *Cangiano* stessa.

Si consideri il pattern: *Cangia.o*, il punto indica la presenza un carattere nella posizione in cui è collocato, per cui tale espressione accerta *Cangiano, Cangiato, Cangia4o* e così via. Si faccia attenzione alla natura case sensitive (di default) dei pattern, per la quale *"cangiano"* con la *'c'* iniziale scritta in *lowercase* (minuscolo) non supererà l'operazione di *match*. Le parentesi quadre *[ ] (brackets)* come accennato nelle tabelle riassuntive, permettono la scelta di un carattere appartenente all'insieme definito.

Il pattern: *[CMK]angiano* risulta valido per *Cangiano, Mangiano, Kangiano* ma non per *angiano, cangiano, mangiano, Vangiano* o *Langiano*, poiché non iniziano per *C, M* o *K* maiuscole. Si noti che *[CMK]* è equivalente a *[C|M|K]*, in quanto il ruolo della barra verticale è quello di presentare una scelta (*C* o *M* o *K*). Il trattino – all'interno delle parentesi quadre permette di definire un range di caratteri consecutivi.

Ad esempio: *[b-z3-9C-P]angiano* è un pattern di regular expression valido per le stringhe aventi come primo carattere una lettera minuscola tra la *b* e la *z* dell'alfabeto inglese, oppure una cifra tra *3* e *9*, o ancora una lettera maiuscola tra *D* e *P*. Quindi si ha un match valido con le stringhe *cangiano, langiano, 3angiano, Cangiano*, ecc… ecc… ma non *aangiano, 2angiano, Tangiano*.

I pattern visti sinora permettono di ricercare in ampi testi tutte le occorrenze delle stringhe adeguate, ma può capitare di dover ricercare una testo che abbia un determinato inizio e una determinata fine.

Due importanti caratteri, molto utilizzati nella realizzazione di pattern reali, sono ^ e *$*. ^ indica l'inizio di una stringa, se ad esempio si ha: ^*Cangiano* la stringa *"Cangiano è l'autore dell'articolo"* è aderente al pattern, ma non a *"L'autore dell'articolo è Cangiano"* perché la parola *Cangiano* deve essere presente all'inizio della stringa. Allo stesso modo, *$* indica la fine di una stringa.

Il pattern: *Cangiano$* sarà coerente con la stringa *"www.visualcsharp.it è il sito di Antonio Cangiano"* ma non con *"Antonio Cangiano è il content manager di www.visualcsharp .it"*, perché la parola *Cangiano* deve essere l'ultima della stringa. Come è facilmente intuibile, l'utilizzo combinato di entrambi i caratteri di posizione, è di ausilio nella definizione univoca delle stringhe.

Si consideri: ^*Cangiano$* la differenza rispetto al primo esempio mostrato sta nel fatto che *"Cangiano"* consente la ricerca di tutte le occorrenze, mentre *"^Cangiano$"* verifica che la stringa sia esattamente

"Cangiano" e non, ad esempio, *"Cangiano Antonio Cangiano".*
Se si dovesse controllare che l'input dell'utente corrisponda ad un particolare codice formato da 3 numeri e una lettera, si potrebbe procedere definendo il seguente pattern: *^[0-9][0-9][0-9][a-zA-Z]$* la presenza dei caratteri di inizio e fine garantisce che sia accettato il codice *680a* ma non *222680a222*. Supponendo che il codice richieda 10 numeri ed un carattere finale, può risultare scomodo dover scrivere *[0-9]* per 10 volte di seguito. A questo proposito esistono dei quantificatori che indicano il numero di occorrenze di una sottoespressione posta precedentemente. Nel caso citato ad esempio potremmo scrivere: *^\d{10}[a-z]$* come visualizzabile nelle tabelle precedenti, \d indica che si tratta di una cifra decimale ed il numero 10 racchiuso tra parentesi graffe, è il quantificatore che indica il numero di ripetizioni dell'espressione antecedente *(\d)*. Altri quantificatori piuttosto comuni sono *?*, *** e *+*.
Vediamo alcuni esempi:

^Cangiano?$ - Indica che una stringa aderente al pattern deve avere *zero* o un carattere *'o'* finale. Quindi *"Cangiano"* e *"Cangianoo"* sono stringhe valide.

^Cangiano+$ - Indica che la stringa aderente al pattern deve avere uno o più caratteri *'o'* finali, per cui *"Cangiano"* non è valida, ma lo è "*Cangianoo"* o *"Cangianoooooooo".*

^Cangiano*$ - L'asterisco indica in questo caso che il carattere *'o'* può essere ripetuto *zero* o più volte. Si noti che la mancanza di parentesi tonde che raggruppano più di un carattere, il quantificatore applica la ripetizione solamente all'ultimo carattere. Se si avesse ad esempio:

^Cangia(no)*$ - la ripetizione sarebbe applicata alle due lette *'no'*. Quindi sarebbero stringhe valide *"Cangia", "Cangiano", "Cangianono", Cangianonono"*, ecc…ecc…
Se si desiderasse ricercare proprio il carattere asterisco, privandolo del suo significato di *"jolly"* basterà anteporre un carattere di backslash \.

Si lascia al lettore la possibilità di approfondire le proprie conoscenze circa le modalità di realizzazione di pattern per la scrittura di regular expressions.

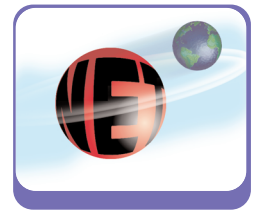
IL PATTERN INIZIALE

Ora che abbiamo posto le basi per la comprensione delle regular expressions passiamo ad una breve analisi del pattern "impressionante" proposto all'inizio dell'articolo:

```
^((4\d{3})|(5[1-5]\d{2})|(6011))-?\d{4}-?\d{4}-?\d{4}|3[4,7]\d{13}$
```

La presenza di un ^ iniziale e di un *$* finale, ci indica che si tratta di una stringa univoca (il numero di carta di credito). All'interno delle parentesi tonde più esterne, sono presenti 3 sottogruppi posti come opzione (o l'uno, o l'altro o l'altro ancora): *(4\d{3}) (5[1-5]\d{2}) (6011)*. Tutti e tre i gruppi accettano 4 caratteri numerici. Il primo è un numero qualsiasi di quattro cifre avente come prima cifra il *4*, il secondo ha un *5* iniziale, una cifra tra *1* e *5* nella seconda posizione, e altri due cifre qualsiasi a completare il quartetto. Il terzo gruppo richiede il numero *6011*. Quindi complessivamente questa prima parte del pattern ci dice che le prime quattro cifre del numero di carta di credito, potrebbe essere *4985, 4222, 4325, 5217, 5399, 5515, 6011*, ecc… ecc… Nella stringa del pattern è poi presente un trattino di separazione seguito da un *?*, che indica che l'utente può o meno inserire un trattino di separazione nel numero di carta. Seguono 4 cifre qualsiasi, un eventuale trattino, altre 4 cifre qualsiasi. Tralasciamo per un momento la parte finale del numero, contenente il formato di un'altra carta di credito. Sinora abbiamo mostrato 3 opzioni, carta Visa da 16 caratteri con prefisso 4, Mastercard 16 caratteri con prefisso 51 o 55, Discover con lunghezza 16 caratteri e prefisso 6011. American Express ha solo 15 caratteri, per questo motivo è l'ultima opzione separata dalle altre; ha come prefisso 34 o 37 ed è seguita da 13 cifre come facilmente deducibile dalla sottostringa *3[4,7]\d{13}*. "L'arcano è svelato", quella espressione totalmente incomprensibile ha ora un senso.

REGEX IN .NET

Mediante la realizzazione di adeguati pattern si possono effettuare manipolazioni testuali di vario tipo, ma per implementare tali operazioni bisogna sfruttare gli strumenti offerti dall'ambiente di sviluppo e il linguaggio prescelto. .NET fornisce il namespace *System.Text* e in particolar modo *System.Text.RegularExpressions* per tali operazioni. Le applicazioni ASP.NET inoltre hanno il controllo *RegularExpressionValidator* per convalidare l'input inserito dagli utenti nelle TextBox delle Webforms. Il *RegularExpressionValidator* ha la proprietà *ValidationExpression* attraverso la quale impostare il pattern di input corretto, in modo da segnalare all'utente eventuali errori e impedire l'invio dei dati. Nella finestra proprietà di Visual Studio, se si fa clic sulla proprietà *ValidationExpression* del controllo, è possibile utilizzare alcuni template offerti dalla finestra di dialogo che appare (mostrata in Fig. 1).
Risulta evidente l'utilità delle regex in ASP.NET per il controllo dell'input utente nei form, ma il vero

☑ SUL WEB
Una delle migliori risorse online per reperire pattern pronti all'uso, utili informazioni e risorse, è il sito:

http://www.regexlib.com

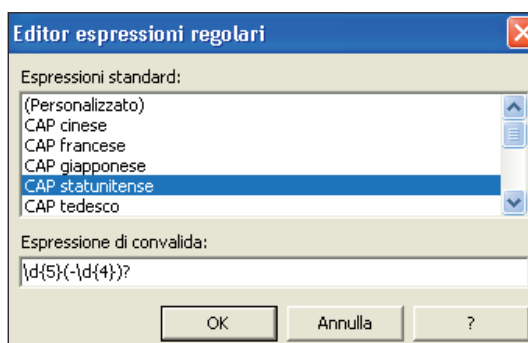
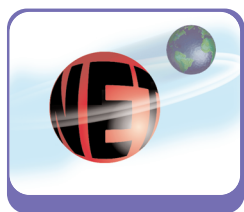



Fig. 1: ***La finestra di dialogo per la scelta di pattern comuni per la proprietà ValidationExpression del RegularExpressionValidator.***

motore delle regex nel framework .NET sono le classi disponibili nel namespace *System.Text.RegularExpressions*. La classe *Regex* è un valido strumento per le operazioni più comuni con le regular expressions. Realizziamo un miniprogramma per verificare l'aderenza di una stringa ad un pattern. Tale programma che può essere realizzato in pochi secondi, può essere utile per la verifica dei primi pattern realizzati nella fase di apprendimento. Avremo bisogno due label e di due textbox (*txtStringa* e *txtPattern*) per permettere all'utente di inserire il testo da verificare e il pattern con il quale controllarlo, ed un pulsante per l'invio (*btnConfronta*).
La form apparirà come mostrato in Fig. 2.

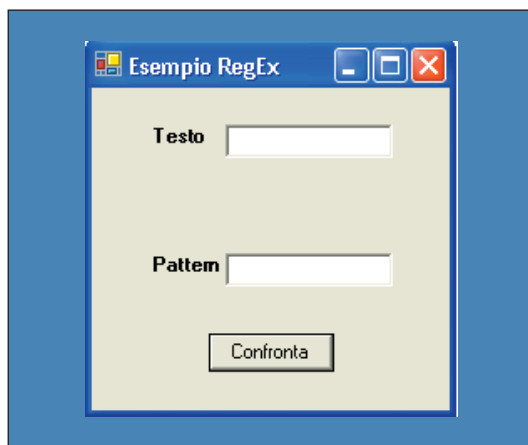



Fig. 2: ***Il semplicissimo layout per la verifica delle Regular Expressions.***

Segue il codice di gestione dell'evento *click* sul button di confronto:

```
private void btnConfronta_Click(object sender,
                              System.EventArgs e) {
  Regex miaRegex = new Regex(txtPattern.Text,
                        RegexOptions.IgnoreCase);
  bool valido = miaRegex.IsMatch(txtStringa.Text);
  if (valido)
    MessageBox.Show("OK! Il testo inserito è valido",
                              "Verifica Regex");
  else
    MessageBox.Show("NO! Il testo inserito non
          corrisponde al pattern", "Verifica Regex"); }
```

La classe *Regex* ha tra gli overloads più utilizzati, il pattern (stringa) e un'opzione che nel nostro caso impone di violare la natura case sensitive delle regular expressions. Il metodo *IsMatch* che riceve come parametro la stringa da verificare, restituisce un valore bool che indica se c'è o meno una corrispondenza tra pattern e testo analizzato.
Un metodo molto utilizzato è *Replace* il cui utilizzo più comune è:

```
string nuovaStringa = Regex.Replace(Stringa, Pattern,
                                    StringaSost)
```

dove ovviamente *StringaSost* è la stringa da sostituire ad ogni occorrenza di testo aderente al pattern. Si supponga ad esempio di voler rendere link tutti gli indirizzi url contenuti in una pagina html.
Basterà qualcosa di analogo a:

```
string Pattern = "(http|ftp|https):
          \/\/[\w]+(.[\w]+)([\w\-\.,@?^=%&amp;
          :/~\+#]*[\w\-\@?^=%&amp;/~\+#])?");
string Stringa = "<p>Ecco alcuni links utili
                    http://www.visualcsharp.it,
                   http://www.windowserver.it,
       http://www.dotnetwork.it, ecc… ecc… </p>"
Regex miaRegex = new Regex(Pattern);
string StringaRisultato = Regex.Replace(Stringa,
                      "<a href=""$0"">$0</a>");
```

ove *$0* rappresenta l'occorrenza aderente al pattern (quindi in questo caso *http://www.visualcsharp.it* e successivamente tutti gli altri url presenti). In fine si noti l'esistenza degli oggetti *Match* e *MatchCollection* utili per ottenere maggiori informazioni circa le occorrenze trovate (*Value, Index, Length*, ecc… maggiori informazioni nella guida in linea o in MSDN).

CONCLUSIONI

La semplicità di utilizzo della classe Regex rende le applicazioni .NET abilitate all'uso di regular expressions nelle più svariate situazioni; come già visto, l'unica difficoltà è la realizzazione delle stesse che, seppur "fastidiose", non spaventano i programmatori più attenti.
Lo scopo dell'articolo non era quello di dare argomentazioni complete, a causa della vastità dell'argomento, ma mostrare una panoramica sulle regular expressions e il loro utilizzo in .NET, con l'intento di incuriosire il lettore, invogliandolo verso uno studio approfondito delle stesse.

Antonio Cangiano

☑ BIBLIOGRAFIA

Due libri particolarmente indicati per approfondire le Regular Expressions ed il loro uso in .NET sono:

- **MASTERING REGULAR EXPRESSIONS (SECOND EDITION)**
  Jeffrey E. Friedl, Andy Oram [O'Reilly]
- **VISUAL BASIC .NET TEXT MANIPULATION HANDBOOK: STRING HANDLING AND REGULAR EXPRESSIONS**
  Paul Wilton, Craig McQueen, François Liger [WROX]

Giocare a briscola in rete con C++ Builder

In queste pagine analizzeremo un'applicazione per giocare una partita a carte in rete.

L'applicazione è stata progettata per il gioco della briscola e come ambiente di sviluppo è stato utilizzato il C++ Builder 6. Il gioco è realizzato con l'ausilio di due applicazioni, client e server, eseguite su computer collegati in rete. Il server resta in ascolto di connessioni e quando il client stabilisce una connessione, il server avvia la partita. Il client, dopo aver impostato l'IP del computer su cui è in esecuzione il lato server, si connette e resta in attesa che il server gli comunichi tutte le carte del mazzo. Dopo aver "distribuito" le carte, si gioca a turno trascinando la carta da giocare su un pannello. L'applicazione invierà la carta giocata all'avversario e resterà in attesa di una risposta. Se si è collegati ad Internet, è possibile inviare dall'applicazione server una e-mail, indirizzata al client, contenente l'indirizzo IP. Da cosa è composta la nostra applicazione:

- **una classe per la gestione del mazzo di carte;**
- **una DLL di risorse contenente le immagini delle carte;**
- **un'applicazione che funge da server;**
- **un'applicazione che funge da client;**
- **un "protocollo" di comunicazione tra server e client.**

DEFINIZIONE DELLA CLASSE

La classe *MazzoDiCarte* (file *ClasseBriscola.cpp*) si occupa della gestione del mazzo di carte. Diamo uno sguardo al file header. Data la scarsa riutilizzabilità della classe si è ritenuto opportuno definire gli attributi pubblici, per non appesantire la classe con metodi di lettura e scrittura di attributi privati.

```
typedef struct {
    int num;
    int seme;
```

```
    int punto;
    String immagine;
}Carta;
class MazzoDiCarte {
  public:
    //attributi
    Carta mazzo[40];  // Mazzo di carte
    Carta briscola;   // carta della briscola
    Carta TreCarteG1[3];  // carte del giocatore sul server
    Carta TreCarteG2[3];  // carte del giocatore sul client
    int turno;        // chi è di turno? Possibili valori: 1,2.
    int puntiG1, puntiG2;  // punteggi dei due giocatori
    //metodi pubblici
    MazzoDiCarte();  //costruttore
    void Inizializza();
    int CalcolaPunti(Carta c1, Carta c2);
    int VincitaMano(Carta C1, Carta C2);
    void DistribuisciCarte();
    //metodi privati
  private:
    void MescolaMazzo(); };
```

La struct *Carta* definisce gli attributi della carta quali il suo valore numerico (da 1 a 10), il seme (da 1 a 4), il punteggio (in base al gioco della briscola) e il nome dell'immagine corrispondente.

I METODI DELLA CLASSE

Il *Costruttore* si occupa di inizializzare il vettore *mazzo* inserendo in esso i valori delle 40 carte. *Inizializza* richiama i metodi *MescolaMazzo()* e *DistribuisciCarte()*. Stabilisce il turno generando un numero casuale. *DistribuisciCarte* distribuisce le carte ai due giocatori valorizzando i vettori *TreCarteG1* e *TreCarteG2*, azzera i punteggi e imposta la briscola. *VincitaMano* riceve due parametri che rappresentano le due carte giocate e determina quale giocatore avrà diritto a giocare per primo al turno successivo (valore restituito). *CalcolaPunti* riceve le due carte,

DLL DI RISORSE
Le immagini delle carte sono contenute in una DLL di risorse che viene caricata dinamicamente all'avvio dell'applicazione. Si poteva inserire la risorsa direttamente nell'applicazione, ma creare una DLL di risorse ha numerosi vantaggi.

somma i punti e restituisce la somma. Definendo virtual i metodi *DistribuisciCarte*, *VincitaMano* e *CalcolaPunti*, sarà possibile derivare una classe per un altro gioco (sarà necessario ridefinire i tre metodi).

PROTOCOLLO DI COMUNICAZIONE

Dovendo far comunicare il server col client (e viceversa) attraverso dei messaggi contenuti in stringhe, è necessario creare un *protocollo di comunicazione*. Le stringhe che vengono inviate nelle due direzioni, contengono come primo carattere un carattere di controllo che corrisponde al tipo di messaggio che un computer invia all'altro. Osserviamo la tabella seguente:

Carattere di controllo	Tipo di Messaggio	Direzione
A	Fine trasmissione carte del mazzo + turno	S->C
B	Carta mazzo numero	S->C
C	Carta mazzo seme	S->C
D	Carta mazzo punto	S->C
E	Carta mazzo immagine	S->C
F	Turno iniziale	S->C
G	Posizione carta giocata dal server (IG1)	S->C
H	Posizione carta giocata dal client (IG2)	C->S
N	Il server si è disconnesso oppure ha chiuso il programma	S->C
M	Il Client si è disconnesso oppure ha chiuso il programma	C->S

Protocollo di trasmissione (Gioco Briscola in Rete)

Nella colonna *direzione* notiamo che la maggior parte delle informazioni viene inviata dal server al client. Il server si occupa della gestione della partita dando inizio ad essa: mescola il mazzo di carte, invia al client tutte le informazioni (4 per ogni carta. Messaggi B,C,D,E) relative al mazzo, invia il turno (messaggio F) e infine invia il messaggio A che indica al client che la partita può iniziare. I messaggi *G* e H indicano che l'avversario ha giocato la sua carta e la comunica all'altro giocatore. I messaggi *N* ed *M* si commentano da soli.
Passiamo ora all'interfaccia visuale.

☑ IL SERVER HA RICEVUTO UNA CARTA

Un messaggio di tipo *'H'* indica che il server ha ricevuto la posizione della carta giocata dal client; la carta viene disegnata sul pannello e viene posto a true il flag *HaGiocatoG2* (indica che il client ha giocato la sua carta). Se il server non ha ancora giocato, vengono attivate le sue carte, infine viene richiamato il metodo *HannoGiocatoG1G2* che si occupa di verificare chi dei due giocatori prende le carte, stabilisce il nuovo turno, aggiorna i punteggi e distribuisce le due nuove carte prelevandole dal mazzo; quando la partita è terminata, emette un messaggio con il nome del vincitore e azzera tutto per una nuova partita. Se il server ha ricevuto un messaggio di tipo *'M'* (client disconnesso) azzera tutto e rimane in ascolto per una nuova connessione. Nel caso in cui i messaggi siano diversi da *'M'* o *'H'* il programma sarà terminato.

L'APPLICAZIONE LATO SERVER

Nel progetto sono state inserite due form: la form principale e la form per l'invio dell'e-mail.

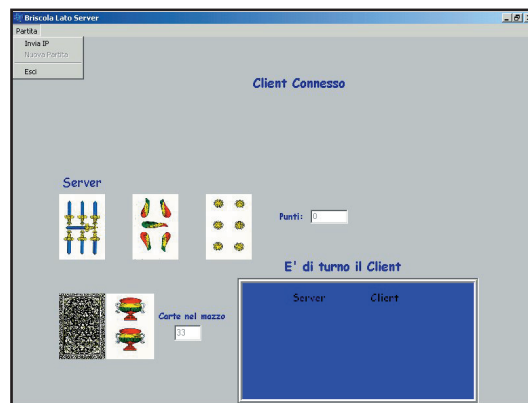



Fig. 1: *Briscola lato server in esecuzione.*

Per motivi di spazio non elencherò qui tutti i componenti utilizzati (e parte del codice), ma potete farvi un'idea aprendo il file *UnitBriscolaServer.dmf* e *InviaEmail.dfm*: avrete modo di vedere i componenti che sono stati utilizzati e le proprietà che sono state modificate in fase di progettazione.

Form di avvio

Di seguito sono riportati gli attributi e i metodi aggiunti alla classe *TForm1* (file *UnitBriscolaServer.h*)

```
private:   // User declarations
   TStringList *Lista; //lista dei messaggi ricevuti
   char CR,LF ;
   void EstraiLista(); //estrae i messaggi dalla lista
   bool EstraiMessaggio(char &,char *);  //decodifica
                         la stringa che contiene il messaggio
    MazzoDiCarte GiocoBriscola;  //istanza
    int carte;  // carte nel mazzo
    bool HaGiocatoG1;  //flag giocatore client
    bool HaGiocatoG2;  //flag giocatore server
    int IG1;  //indice della carta giocata dal client
    int IG2;  //indice della carta giocata dal server
    int treCarte;  //ultime tre carte
    int StatoConnessione; // 0 In attesa di connessione
    // 1 Connesso
    //-1 Disconnesso
    // 2 Connesso e pronto per giocare
    void TrasmettiCarte(MazzoDiCarte);
    void TrasmettiCartaGiocata(int);
    void HannoGiocatoG1G2();
    void AzzeraTutto();
public:    // User declarations
    __fastcall TForm1(TComponent* Owner);
```

Per consentire la comunicazione in rete è stata inserita nella form principale un componente *ServerSocket*. Questo componente ci permette di inviare e ricevere informazioni da un altro computer collegato in rete senza dover ricorrere alle API. Il *ServerSocket* apre la porta di servizio 6711 e resta in ascolto di connessioni TCP/IP. Se si vuole utilizzare un'altra

porta sarà necessario assicurarsi che questa non sia associata ad un servizio noto oppure non sia utilizzata da un'altra applicazione. Nell'evento *FormCreate* troviamo le seguenti righe di codice:

```
dllInstance=LoadLibrary("PrgDLL.dll");
    ServerSocket1->Active=true;
```

La prima carica la libreria di risorse che contiene le immagini, la seconda attiva il componente *ServerSocket* e lo pone in ascolto sulla porta. Di seguito è riportato il codice dell'evento *FormShow* che mostra come caricare l'immagine dalla risorsa.

```
void __fastcall TForm1::FormShow(TObject *Sender) {
  // Carica l'immagine della carta dalla DLL
  Image7->Picture->Bitmap->LoadFromResourceName(
                        (int)dllInstance,"ROVESCIO");
  Edit1->Text="40";
}
```

Quando il client stabilisce una connessione, viene invocato il metodo *ServerSocket1ClientConnect*.

```
void __fastcall TForm1::ServerSocket1ClientConnect(
      TObject *Sender, TCustomWinSocket *Socket) {
  lblStatoConnessione->Caption="Client Connesso";
    StatoConnessione=1;
    mnuItemNuovaPartita->Enabled=true;
    mnuItemInvia->Enabled=false; }
```

Il gestore *ServerSocket1ClientConnect* aggiorna la label che ci informa sullo stato della connessione e attiva la voce *"Nuova partita"* del menu. La partita può iniziare.
Il gestore dell'evento click di *"Nuova Partita"* azzera tutte le variabili, invoca il metodo *Inizializza* dell'istanza *GiocoBriscola*, carica le immagini delle tre carte del giocatore e della briscola, attiva le tre carte se è di turno il server e richiama il metodo *TrasmettiCarte*

```
void  TForm1::TrasmettiCarte(MazzoDiCarte Gioco)
{}
```

Per ogni informazione da inviare, viene creata una stringa contenente come primo carattere il carattere di controllo, l'informazione e infine la coppia di caratteri CR LF. Col metodo *SendText* la stringa viene inviata al client. Se è di turno il client, il server resta in attesa della carta giocata dall'avversario.
Quando arriva un messaggio dal client, viene invocato il gestore *ServerSocketClientRead*

```
void __fastcall TForm1::ServerSocket1ClientRead(
      TObject *Sender, TCustomWinSocket *Socket) {
    Lista->Text= Socket->ReceiveText();
    EstraiLista(); }
```

Per gestire la lista dei messaggi è stato definito un attributo *TstringList *Lista* a cui viene assegnata la stringa ricevuta.
I messaggi vengono poi elaborati dalla funzione *EstraiLista*, che estrae dalla *Lista* un messaggio per volta e, tramite la funzione *EstraiMessaggio*, ne decodifica il contenuto, estraendo il carattere di controllo e il contenuto del messaggio.

```
void  TForm1::EstraiLista()
{}
bool TForm1::EstraiMessaggio(char &p,char
                                  *stringaMessaggio)
{}
```

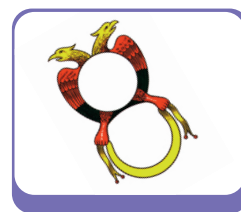
INVIARE IL PROPRIO IP PER EMAIL

La Form per l'invio del proprio IP (*unit InviaEmail*) si attiva dalla form principale selezionando la voce *"Invia IP"* del menu. Per poter inviare email, C++ Builder ci fornisce il componente *NMSMTP*. La Edit che contiene l'IP è di sola lettura perché l'IP viene determinato dal programma lanciando il comando dos *ipconfig*. Inseriti tutti i dati nelle altre edit, si invia il messaggio cliccando sul tasto *Invia IP*. Una *MessageBox* ci avviserà che il messaggio è stato inoltrato e possiamo ritornare alla form principale. Accertatevi che il server SMTP accetti connessioni sulla porta 25 (alcuni server applicano restrizioni di sicurezza che possono bloccare l'invio di messaggi).

Membri aggiunti alla classe *TformInviaMail* (file *InviaEmail.h*)

```
private:   // User declarations
  void EstraiIPdalFile();
public:                  // User declarations
  bool MsgInviato;
  __fastcall TFormInviaMail(TComponent* Owner);
```

L'attributo *MsgInviato* sarà testato dalla form principale per verificare se l'e-mail è stata inviata. Quando il form si apre, viene richiamato il gestore *FormShow* che esegue il comando DOS *ipconfig* inviando l'output sul file *Log_IP.txt*; estrae poi dal file l'IP, tramite la funzione *EstraiIPdalFile* e lo copia nella *EditIP*. Prima di uscire cancella il file *Log_IP.txt*.

```
void __fastcall TFormInviaMail::FormShow(TObject
                                        *Sender) {
   char NomeFile[15]="Log_IP.txt";
   char ComandoDos[50]="ipconfig > ";
   strcat(ComandoDos,NomeFile);
   system(ComandoDos);  // esegue il comando DOS
   EstraiIPdalFile();
// cancella il file Log_IP.txt
```

☑ È DI TURNO IL SERVER

Quando il server è di turno, può giocare la sua carta trascinandola nel pannello. Per il trascinamento delle carte sono stati attivati due eventi: *Image1EndDrag* e *Panel1DragDrop*. *Panel1DragDrop* assegna al componente *Image9*, che rappresenta la carta giocata dal server, l'immagine della carta che è stata trascinata sul pannello. Quando nel trascinamento rilasciamo il tasto sinistro del mouse e la carta è sul Panel, si attiva l'evento *Image1EndDrag*, che pone a *false* la proprietà *Visibile* della carta giocata, trasmette la sua posizione al client richiamando la funzione *TrasmettiCarta*, pone a *true* il flag *HaGiocatoG1* e disabilita le carte. Prima di uscire dalla funzione richiama il metodo *HannoGiocatoG1G2*. Anche *Image2* e *Image3* abilitano l'evento *Image1EndDrag*.

```
void __fastcall
 TForm1::Panel1DragDrop
         (TObject *Sender,
          TObject *Source,
               int X, int Y)
{}
```

☑ IL CLIENT SI È DISCONNESSO

Il gestore di eventi *ServerSocketClientDisconnect* viene richiamato quando il client si disconnette oppure chiude il programma. Il socket ritorna in stato di ascolto in attesa di nuove connessioni.

☑ LOCALIZZARE UNA APPLICAZIONE

Supponiamo di voler distribuire l'applicazione in un Paese in cui per giocare a carte si utilizzi un mazzo con immagini diverse dalle carte napoletane. Possiamo inserire le immagini delle nuove carte in un'altra DLL di risorse (mantenendo invariati i nomi delle immagini nella risorsa) e chiedere all'utente, all'avvio dell'applicazione, di scegliere il mazzo di carte che vuole utilizzare. Essendo le DLL caricate dinamicamente, è possibile richiamare quella corrispondente alla scelta fatta dall'utente senza dover produrre due applicazioni diverse per ogni mazzo di carte. Si può fare la stessa cosa per le didascalie dei componenti visuali dell'applicazione, inserendo in una risorsa le stringhe delle didascalie tradotte nella lingua locale.

```
strcpy(ComandoDos,"del ");
strcat(ComandoDos,NomeFile);
system(ComandoDos); }
void TFormInviaMail::EstraiIPdalFile()
{}
```

Il gestore *btnInviaClick* verifica che i campi per l'invio della e-mail non siano vuoti e imposta tutte le proprietà del componente SMTP:

```
NMSMTP1->TimeOut=30000;
NMSMTP1->Host=EtSMTP->Text;
NMSMTP1->PostMessage-
                    >FromAddress=EtEmailMittente->Text;
NMSMTP1->PostMessage->FromName=
                              EtNomeMittente->Text;
NMSMTP1->PostMessage->ToAddress->Text=
                              EtEmailDestinatario->Text;
NMSMTP1->PostMessage->Body->Text=EtIP->
                    Text+String(CR)+String(Date())+
       String(CR)+String(Time());
NMSMTP1->PostMessage->Subject="Briscola";
```

Invia l'email utilizzando i seguenti metodi:

```
NMSMTP1->Connect();
NMSMTP1->SendMail();
NMSMTP1->Disconnect();
```

Nel corpo del messaggio vengono memorizzati l'IP, la data e l'ora di invio; mentre nella proprietà *Object* è memorizzata la stringa *"Briscola"* che permetterà al client di identificare il messaggio. Se il messaggio viene inviato, si attiva l'evento *OnSuccess* di *NMSMTP*, che pone a *true* il flag *MsgInviato* ed emette un messaggio per avvisare l'utente che l'e-mail è stata inviata. In caso di insuccesso, si attiva l'evento *OnFailure* e il flag *MsgInviato* viene posto a *false*. Inviato il messaggio, si torna alla form principale e si resta in attesa di una connessione dal client.

L'APPLICAZIONE LATO CLIENT

Gran parte del codice, soprattutto per quel che riguarda la gestione delle carte è comune alle due applicazioni. Quindi la descrizione di questa parte sarà più breve. All'avvio del lato client, si può leggere la posta per verificare se nella casella di posta c'è una mail con oggetto *"Briscola"* (in questo caso l'IP letto dalla mail sarà scritto nella casella di testo *EditIP*) oppure, se si conosce già l'IP del computer su cui è installato il lato server, lo si può inserire direttamente nella *EditIP*. Impostato l'IP, ci si connette cliccando sulla voce *Connetti* del menu.
Se il server è in ascolto, il client cambia la caption

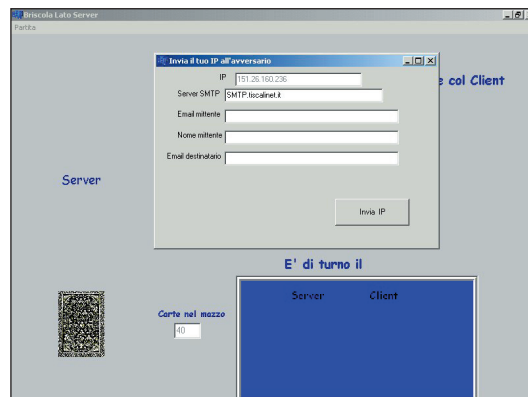



Fig. 2: Invio dell'IP tramite email.

della label che ci informa sullo stato della connessione e resta in attesa di ricevere le carte. Anche in questo progetto sono state inserite due form: la form principale e la form per la ricezione dell'email. Consultate i file *UnitBriscolaClient.dmf* e *RiceviEmail.dfm* per vedere i componenti che sono stati utilizzati e le proprietà che sono state modificate in fase di progettazione.

Form di avvio

Per comunicare col server è stato inserito nella form principale un componente *ClientSocket*. La porta utilizzata deve essere la stessa definita nel lato server. Questo componente viene attivato quando si clicca sulla voce *Connetti* del menu. Prima di attivarlo è necessario impostare la proprietà *ClientSocket1->Address* assegnandole l'indirizzo IP del server.

```
ClientSocket1->Address=EditIP->Text;
ClientSocket1->Active=true;
```

Stabilita la connessione tra i due computer, il client resta in attesa di ricevere messaggi. L'evento *ClientSocketRead* si attiva quando il server invia un messaggio

```
void __fastcall TForm1::ClientSocket1Read(TObject
                                          *Sender,
 TCustomWinSocket *Socket) {
     String s;
     s=Socket->ReceiveText();
     Lista->Text=s;
     EstraiLista(); }
```

Osserviamo la funzione *EstraiLista* presente in *UnitBriscolaClient.cpp*:

```
void  TForm1::EstraiLista()
{}
```

Esaminiamo i tipi di comandi:

Comando B, C, D, E: il client ha ricevuto una carta

e la inserisce nel mazzo.
Comando F: aggiorna il turno.
Comando N: il server si è disconnesso; viene aggiornato tutto per preparare il client ad un'altra connessione.
Comando A: richiama il metodo *NuovaPartita* che distribuisce le carte, le visualizza, aggiorna tutte le variabili e, se è di turno il client, abilita le sue carte. La partita può iniziare.
Comando G: il client ha ricevuto la posizione della carta giocata dal server.

La gestione della partita è identica al lato server. Per la trasmissione della carta giocata è stato definito il metodo *TrasmettiCartaGiocata*, che utilizza il metodo *SendText* del Socket per l'invio del messaggio

```
void  TForm1::TrasmettiCartaGiocata(int G2) {
      String s;
      Cursor=crHourGlass;
      s="H"+String(G2)+String(CR)+String(LF);
      ClientSocket1->Socket->SendText(s);
      Cursor=crDefault; }
```

La voce *"Ricevi IP"* del menu apre la form per la ricezione della email e imposta l'IP.

Form RiceviEmail
Dopo aver inserito i dati nelle Edit *Server POP, User ID* e *Password*, si può cliccare sul tasto *"C'è nessuno?"*. Il gestore dell'evento associato a questo tasto preleva, senza scaricarla, la posta; analizza per ogni messaggio l'object e se lo trova uguale alla stringa *"Briscola"*, scarica il messaggio e ne estrae dal corpo l'IP, la data e l'ora di invio. Se però, nel frattempo, chi ci ha spedito il messaggio si è disconnesso dalla rete Internet, l'indirizzo IP che leggiamo non è più associato a quell'utente. Perciò, prima di connetterci, diamo uno sguardo alla data e l'ora di invio del messaggio. Chiudiamo questa finestra per tornare alla form principale, dove troveremo l'IP ricopiato nella *EditIP.* Possiamo connetterci al server cliccando sulla voce *"Connetti"* del menu. Nel form sono definite tre Edit per inserire i dati (*Indirizzo server POP, User ID* e *Password*). Le altre tre Edit sono di sola lettura e conterranno i dati ricevuti nella email. Il componente che ci permette di leggere la posta dalla nostra casella sul server POP è *NMPOP3*. Le sue proprietà vengono modificate in fase di esecuzione, quando si clicca sul tasto *"C'è nessuno?"*.

```
NMPOP31->TimeOut=30000;
NMPOP31->Host=EtIndirizzoPOP->Text;
NMPOP31->UserID=EtUserID->Text;
NMPOP31->Password=EtPassword->Text;
```

Il metodo *GetMailMessage* permette di prelevare la email:

```
NMPOP31->DeleteOnRead=false;  //non scaricare la posta
NMPOP31->Connect();
for(int i=0; i<NMPOP31->MailCount; i++){
      NMPOP31->GetMailMessage(i+1);
    if(NMPOP31->MailMessage->Subject==" Briscola")
    { NMPOP31->DeleteOnRead=true;  //scarica la
                                        email (i+1)
      NMPOP31->GetMailMessage(i+1);
      NMPOP31->DeleteOnRead=false;
      EtIP->Text=NMPOP31->MailMessage->Body->
                              Strings[1]; //preleva l'IP
      EtData->Text=NMPOP31->MailMessage->Body->
                          Strings[2]; //preleva la data
      EtTime->Text=NMPOP31->MailMessage->Body->
                           Strings[3];//preleva l'ora
      break;}
}
NMPOP31->Disconnect();
CancellaTMP();
```

Dopo esserci disconnessi dal server POP dobbiamo eliminare dalla cartella, in cui è in esecuzione, il programma, tutti i file temporanei che si sono creati durante la lettura dei messaggi. Questa operazione è affidata alla funzione *CancellaTMP.*

```
void  TFormRiceviIP::CancellaTMP() {
 // calcella tutti i file temporanei che sono stati creati
 //   durante la lettura della posta
    char ComandoDos[50]="del *.tmp";
    system(ComandoDos);
    strcpy(ComandoDos,"del *.jpg");
    system(ComandoDos);
    strcpy(ComandoDos,"del *.mme");
    system(ComandoDos);
}
```

Il codice di quest'ultima funzione elimina tutti i file con estensione tmp, jpg e mme che trova nella cartella in cui è installato il programma. Se eseguite l'applicazione all'esterno della sua cartella rischiate di perdere file che possono esservi utili.

DOVEROSE RACCOMANDAZIONI

Concludiamo rivolgendo una doverosa raccomandazione a chi esegue il lato server dell'applicazione: conoscete bene il vostro avversario? La conoscenza del vostro indirizzo IP potrebbe stuzzicare la sua "anima hacker" (o cracker) e, mentre giocate, lui potrebbe interrogare il vostro computer per sapere se ci sono altre porte aperte e altro. Se non vi fidate del vostro avversario e il vostro computer non è protetto da un firewall, sarà meglio rinunciare alla briscola e farsi un tranquillo solitario.

Patrizia Martemucci

☑ CREARE UNA DLL DI RISORSE

Armatevi di pazienza e passate allo scanner le 40 carte più una carta rovesciata che rappresenterà il mazzo e salvate ognuna in un file bmp (scegliete risoluzioni con massimo 256 colori). Richiamate Image Editor (tool fornito da Borland con C++ Builder) e create un nuovo file di risorse (.res). Inserite nella nuova risorsa tante nuove Bitmap quante sono le carte e ricopiate in ognuna le immagini salvate precedentemente. Salvate la risorsa e aprite C++ Builder. Create un nuovo progetto per DLL e attraverso il Project Manager inserite nel progetto la risorsa. Compilate il progetto senza modificare il codice e la DLL di risorse sarà già pronta per l'uso (la compilazione produce un file con estensione DLL). Una volta creata la DLL possiamo eliminare tutti i file che abbiamo utilizzato per la sua creazione.

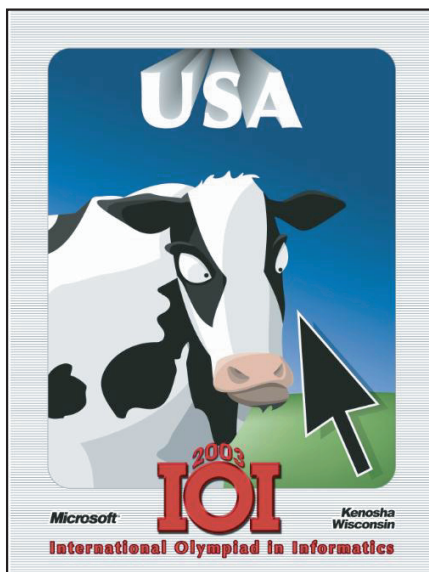
Olimpiadi Internazionali dell'informatica

Caccia all'oro

Le olimpiadi Internazionali dell'informatica 2003 hanno portato all'Italia un medagliere piuttosto ricco, con due argenti e un bronzo.

Giunte alla loro quindicesima edizione, le Olimpiadi Internazionali dell'Informatica 2003 (IOI), svoltesi alla University of Wisconsin dal 16 al 23 agosto, sono solo una delle sei Olimpiadi "Scientifiche" che ogni anno coinvolgono gli studenti delle scuole superiori di numerose nazioni del mondo. Le IOI ormai da 15 anni affiancano le Olimpiadi di Matematica, Fisica, Chimica, Biologia ed Astronomia (le ultime arrivate). Quella che si è svolta quest'anno negli Stati Uniti, alla University of Wisconsin dal 16 al 23 agosto è stata la 15ª edizione e ha visto la partecipazione di 81 nazioni, con l'Italia impegnata a recitare un ruolo importante, conquistando tre medaglie, due argenti e un bronzo; un buon auspicio per la prossima edizione, con l'inevitabile speranza che la prossima sia una medaglia d'oro.

NON SOLO VIDEOGAME

Lo scopo della manifestazione, patrocinata dall'UNESCO e promossa in Italia dal MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca) e dall'AICA (Associazione Italiana per l'Informatica e il Calcolo Automatico), è ovviamente quello di stimolare l'interesse dei giovani nei confronti degli aspetti scientifici e culturali relativi alle tecnologie informatiche. Per una volta quindi lo scopo è quello di appassionare i ragazzi agli elementi di logica e ragionamento che stanno alla base della programmazione, lasciando da parte gli aspetti videoludici grazie ai quali i Pc sono tanto popolari tra i più giovani. Ma le IOI non sono solo "informatica e programmazione" e, infatti, durante la settimana in cui si sono svolte le IOI, solo due erano i giorni impegnati con le prove, mentre i restanti 5 giorni sono stati dedicati a momenti di aggregazione e socializzazione tra i ragazzi partecipanti, con diverse attività ricreative.

LA STRADA PER LA VITTORIA

Il regolamento è piuttosto semplice: ogni nazione seleziona e porta alle Olimpiadi 4 studenti scelti attraverso una competizione nazionale. Una volta arrivato alle Olimpiadi, ogni ragazzo compete in modo individuale affrontando diversi problemi di natura algoritmica, da risolvere utilizzando un linguaggio di programmazione a scelta tra C/C++ e Pascal (ma il linguaggio può variare tra un'edizione e l'altra delle IOI). Per ognuno dei due giorni, di gara i ragazzi devono affrontare 3 problemi avendo a disposizione un tempo giornaliero di 5 ore. Per ogni esercizio ricevono quindi un punteggio, e la somma dei risultati ottenuti nelle due giornate di gara determina il piazzamento in classifica e, di conseguenza, la distribuzione delle medaglie.
Ovviamente affinché la prova dia esito positivo, il PC su cui il concorrente ha compilato il programma dovrà essere in grado di dare la risposta al quesito proposto, ma non solo, dovrà farlo nel tempo massimo stabilito dal testo del problema. Le premiazioni avvengono per fasce di medaglie e quest'anno ci sono state 24

medaglie d'oro, 45 argenti e 63 bronzi, mentre il titolo di campione è stato conquistato da Hwan-Seung Yeo, rappresentante della squadra coreana che con il punteggio di 455,4 ha staccato di quasi 30 punti la seconda medaglia d'oro, proveniente dalla Bulgaria. La Corea, insieme ad altri paesi dell'est e dell'Europa orientale, si è confermata la "nazione da battere", portando nei primissimi posti tutti i suoi rappresentanti, conquistando due ori e due argenti.

LA STRADA PER IL WINSCONSIN

Per l'Italia tutto è cominciato verso la fine del 2002, quando più di 360 istituti, con oltre 7500 studenti, hanno risposto al bando di concorso per le Olimpiadi Internazionali dell'Informatica 2003. Le prime, pesanti, selezioni interne alle scuole hanno portato 82 candidati a disputare la selezione nazionale (da quest'anno Olimpiadi Italiane di Informatica), da cui sono usciti i 21 partecipanti meglio piazzati che hanno seguito un corso di formazione presso l'Università di Pisa. Tra i 21 finalisti sono quindi stati selezionati i 4 titolari e le due riserve per il team italiano, premiati per il bel risultato con un computer portatile. I nostri portacolori hanno ottenuto un ottimo risultato, ma la differenza di prestazione, per esempio, nei confronti della nazionale coreana, testimonia il ritardo che il nostro sistema scolastico paga in campo informatico rispetto a molti paesi dell'est, asiatico e europeo. I quattro ragazzi della nostra nazionale sono, prima di tutto, appassionati di informatica e solo due hanno frequentato scuole con una specializzazione legata alla programmazione, mentre i due restanti hanno conseguito un diploma di liceo scientifico. Impossibile, quindi, competere con i coreani sul piano della formazione scolastica, anche se molto è stato fatto in fase di preparazione proprio durante le settimane di allenamento che hanno preceduto la partenza per gli Stati Uniti. Per la prossima edizione, che verrà ospitata dalla Grecia, sono già in fase avanza i preparativi. Tutte le informazioni per la partecipazione all'edizione 2004 (il bando si chiuderà il 31 ottobre), sono reperibili sul sito *www.olimpiadi-informatica.it* e chissà che questa volta, con il ritorno in Europa e con un maggior numero di partecipanti alle selezioni, dopo i bronzi e gli argenti delle stagioni passate, non ci scappi una bella medaglia d'oro.

Stefano Tura

☑ LA SQUADRA AZZURRA

A difendere i colori dell'Italia hanno provveduto, con ottimi risultati, quattro giovanissimi ragazzi.

Dal racconto di tutti è emerso l'entusiasmo per l'esperienza vissuta, a contatto, per una volta, con persone che non ti guardano storto solo perché trovi appassionante l'informatica piuttosto che i videogame. Il bello di condividere una passione non ha però messo da parte lo spirito di rivalità che, una volta davanti al compilatore, è stato fortissimo facendo emergere la soddisfazione di chi ha raggiunto il risultato migliore e l'amarezza (e la rabbia) di chi quest'anno ha dovuto rinunciare alla medaglia. Su una cosa però tutti sono stati d'accordo: troppo poche le ragazze (4 o 5) e solo una piuttosto carina, la cinese, ma questa è un'altra storia.

GILBERTO ABRAM

Vive a: Ronzone (TN)
Età: 19 anni
Studi: Diploma di Liceo Scientifico
Numero di Olimpiadi: 1
Miglior piazzamento: Medaglia d'argento
Hobby: Chitarra, musica punk

MATTEO BRUNI

Vive a: Monte San Martino (MC)
Età: 19 anni
Studi: ITIS – Specializzazione Informatica
Numero di Olimpiadi: 1
Miglior piazzamento: Medaglia di bronzo
Hobby: Bicicletta, tennis tavolo, calcio

GIUSEPPE OTTAVIANO

Vive a: Ragusa
Età: 18 anni
Studi: Diploma di Liceo Scientifico
Numero di Olimpiadi: 1
Miglior piazzamento: Medaglia d'argento
Hobby: Musica classica, rock, punk

STEFANO MAGGIOLO

Vive a: Abano (PD)
Età: 18 anni
Studi: ITIS – Specializzazione informatica
Numero di Olimpiadi: 2
Miglior piazzamento: Medaglia di bronzo, Medaglia d'oro alle Olimpiadi Italiane
Hobby: Tennis tavolo a livello agonistico, musica

I trucchi del mestiere

Tips&Tricks

La rubrica raccoglie trucchi e piccoli pezzi di codice che solitamente non trovano posto nei manuali, ma sono frutto dell'esperienza di chi programma. Alcuni trucchi sono proposti dalla Redazione, altri provengono da una ricerca su internet, altri ancora ci giungono dai lettori. Chi vuole contribuire potrà inviarci i suoi tips&tricks preferiti che, una volta scelti, verranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

UN DOWNLOAD MANAGER

Spesso è capitato di progettare applicazioni che necessitano di interagire con internet per il download di file, purtroppo non tutti hanno a propria disposizione una connessione a banda larga e, di conseguenza, quando le dimensioni dei download superano qualche mega, il tempo necessario per completare l'operazione non si riduce a pochi minuti, senza contare che si potrebbero avere delle disconnessioni accidentali, e se non si utilizzano programmi tipo *"GetRight"*, tutto il lavoro di download potrebbe andare perso.
La classe qui presentata permette di downloadare file, con la possibilità di fermare l'operazione di download in qualsiasi momento, e di riprendere l'operazione in seguito; altresì, se accidentalmente si perde la connessione, tutto ciò che si è scaricato non viene perso, poiché viene effettuare una copia dei file nella cartella temporanea denominata *"C:\Down_"*.
Trovate il progetto completo nella directory tips del Cd-Rom allegato o sul Web: *cdrom.ioprogrammo.it*

Tip fornito dal sig. P .Libro

COME GESTIRE LA TASKBAR

Una serie di funzioni utili a visualizzare e a gestire un'icona nella task bar. Il tutto funziona tramite la chiamata alla funzione API *"Shell_NotifyIcon"* della libreria *"shell32"* e alla dichiarazione del tipo dato *"NOTIFYICONDATA"*.

Tip fornito dal sig. M. Garbujo

```
'Dichiarazione della struttura per la gestione dell'Icona di notifica
Private Type NOTIFYICONDATA
  cbSize As Long
  hWnd As Long
  uId As Long
  uFlags As Long
  uCallBackMessage As Long
  hIcon As Long
  szTip As String * 64
End Type
```

```
'Comandi per l'inserimento, la modifica e cancellazione dell'Icona
                                                          di notifica
Private Const NIM_ADD = &H0
Private Const NIM_MODIFY = &H1
Private Const NIM_DELETE = &H2

'Ritorno informazioni del mouse integrate con l'Icona di Notifica
Private Const WM_MOUSEMOVE = &H200
Private Const WM_LBUTTONDOWN = &H201     'Pulsante sinistro giù
Private Const WM_LBUTTONUP = &H202      'Pulsante sinistro su
Private Const WM_LBUTTONDBLCLK = &H203   'Doppio click
                                                   pulsante sinistro
Private Const WM_RBUTTONDOWN = &H204     'Pulsante destro giù
Private Const WM_RBUTTONUP = &H205      'Pulsante destro su
Private Const WM_RBUTTONDBLCLK = &H206   'Doppio click
                                                     pulsante destro

'Valori dei flag per attivare il messaggio, l'icona ed il tooltip
Private Const NIF_MESSAGE = &H1
Private Const NIF_ICON = &H2
Private Const NIF_TIP = &H4

'Dichiarazione della chiamata alla funzione API
Private Declare Function Shell_NotifyIcon Lib "shell32" Alias
            "Shell_NotifyIconA" (ByVal dwMessage As Long, pnid As
                                      NOTIFYICONDATA) As Boolean

Dim nid As NOTIFYICONDATA

Public Sub AddIconaNotifica()
  nid.cbSize = Len(nid)
  nid.hWnd = Form1.hWnd
  nid.uId = vbNull
  nid.uFlags = NIF_ICON Or NIF_TIP Or NIF_MESSAGE
  nid.uCallBackMessage = WM_MOUSEMOVE
  nid.hIcon = Form1.Icon
  nid.szTip = "Icona di notifica" & vbNullChar

  Shell_NotifyIcon NIM_ADD, nid
End Sub

Private Sub DeleteIconaNotifica()
  Shell_NotifyIcon NIM_DELETE, nid
End Sub
Private Sub Form_MouseMove(Button As Integer, Shift As Integer,
```

IL TIP DEL MESE

UNA FUNZIONE PER CHIAMARE UN NUMERO TELEFONICO

Spesso, per comporre un numero telefonico da Visual Basic, si ricorre al componente OCX *MSCOMM32* che, se opportunamente configurato, consente di gestire il modem per comporre un numero telefonico; tuttavia, esiste una particolare API, *tapiRequestMakeCall*, che permette di effettuare la medesima operazione in modo molto più semplice e immediato.

Tip fornito dalla Sig.ra N. Martire

```
Option Explicit

Private Declare Function tapiRequestMakeCall& Lib "TAPI32.DLL"
        (ByVal DestAddress$, ByVal AppName$, ByVal CalledParty$,
                                                ByVal Comment$)
Private Const TAPIERR_NOREQUESTRECIPIENT = -2&
Private Const TAPIERR_REQUESTQUEUEFULL = -3&
Private Const TAPIERR_INVALDESTADDRESS = -4&

Private Sub cmdChiama_Click()
    Dim Strtemp As String
    Dim nResult As Long

    nResult = tapiRequestMakeCall&(Trim$(txtNumero),
                                CStr(Caption), "Test telefono", "")
    If nResult <> 0 Then
        Strtemp = "Errore durante la composizione del numero: "
        Select Case nResult
            Case TAPIERR_NOREQUESTRECIPIENT
                Strtemp = Strtemp & "Non è possibile avviare
                         l'applicazione Windows si supporto"
            Case TAPIERR_REQUESTQUEUEFULL
                Strtemp = Strtemp & "Il sistema è attualmente
                  impegnato in altre composizioni telefoniche"
            Case TAPIERR_INVALDESTADDRESS
                Strtemp = Strtemp & "Il numero telefonico non è corretto"
            Case Else
                Strtemp = Strtemp & "Errore sconosciuto"
        End Select
        MsgBox Strtemp
    End If

End Sub

Private Sub Form_Load()
    Abilita_Chiamata
End Sub

Private Sub cmdExit_Click()
    Unload Me
End Sub

Private Sub Abilita_Chiamata()
    cmdChiama.Enabled = Len(Trim$(txtNumero)) > 0
End Sub

Private Sub txtNumero_Change()
    Abilita_Chiamata
End Sub
```

```
                                        X As Single, Y As Single)
    Dim msg As Long
    Dim sToolTip As String

    msg = X / Screen.TwipsPerPixelX
    Select Case msg
        Case WM_LBUTTONDOWN
        Case WM_LBUTTONUP
        Case WM_LBUTTONDBLCLK
            MsgBox "E' stato premuto il tasto sinistro"
        Case WM_RBUTTONDOWN
            sToolTip = InputBox("Inserire il nuovo tooltip:", "Conferma")
            If sToolTip <> "" Then
                nid.szTip = sToolTip & vbNullChar
                Shell_NotifyIcon NIM_MODIFY, nid
            End If

        Case WM_RBUTTONUP
        Case WM_RBUTTONDBLCLK
    End Select
End Sub
```

COME CREARE UN'ICONA PARTENDO DA UN'IMMAGINE

Semplice ma funzionale! Poche righe di codice Visual Basic .NET per creare un'icona da un'immagine JPG o BMP.
Nell'esempio, al click di un bottone,denominato *button1*, viene richiamata una procedura *(Crea_Icona)* che trasforma l'immagine *web.jpg* (contenuta nel medesimo percorso dell'applicazione) in una classica icona Windows.

Tip fornito dal Sig. E. Mattei

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
                         System.EventArgs) Handles Button1.Click
    Crea_Icona(Directory.GetCurrentDirectory & "\web.jpg")
End Sub
Public Sub Crea_Icona(ByVal StrPercorso As String)
```

```
    Dim bmp As New Bitmap(StrPercorso)
    Dim img As IntPtr = bmp.GetHicon
    Dim ico As Icon = Icon.FromHandle(img)
    Dim fStream As New FileStream(Directory.GetCurrentDirectory &
                              "\web.ico", FileMode.Create)
    ico.Save(fStream)
    fStream.Flush()
    fStream.Close()
    MsgBox("Icona Creata", MsgBoxStyle.Information, "Creazione
                                                        Icone")
End Sub
```

DELPHI

COME RECUPERARE UN FILE XML DAL WEB

Il tip consente di agganciarsi ad un indirizzo Web e downloadare, in locale, il contenuto di un documento XML; i dati presenti in quest'ultimo saranno quindi formattati e inseriti in una ListBox appositamente configurata. Nell'esempio, il sito web dal quale reperire il file xml è segnato come *http://www.localhost*, mentre il file XML è identificato come *index.xml*.

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
          Forms, Dialogs, xmldom, XMLDoc, msxmldom, ComCtrls,
                              StdCtrls, ExtCtrls, XMLIntf;
type
  TForm1 = class(TForm)
    lv: TListView;
    XMLDoc: TXMLDocument;
    pnlTop: TPanel;
    btnRefresh: TButton;
    procedure btnRefreshClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
uses ExtActns;
function DownloadURLFile(const URL_XML, File_Locale : TFileName) :
                                                        boolean;
begin
  Result:=True;
  with TDownLoadURL.Create(nil) do
  try
    URL:=URL_XML;
    Filename:=File_Locale;
    try
      ExecuteTarget(nil);
    except
      Result:=False;
    end;
  finally
    Free;
  end;
end;

procedure TForm1.btnRefreshClick(Sender: TObject);
const
  URL_XML = 'http://localhost/index.xml';
var
  File_Locale : TFileName;

  StartItemNode : IXMLNode;
  ANode : IXMLNode;
  STitle, sDesc : widestring;
begin
 File_Locale := IncludeTrailingPathDelimiter(ExtractFilePath(
                Application.ExeName)) + 'temp.adpheadlines.xml';

  Screen.Cursor:=crHourglass;
  try
    if not DownloadURLFile(URL_XML, File_Locale)  then
    begin
      Screen.Cursor:=crDefault;
      Raise Exception.CreateFmt('Non è stato possibile connettersi
                                                   al Web',[]);
      Exit;
    end;
    if not FileExists(File_Locale) then
    begin
      Screen.Cursor:=crDefault;
      raise exception.Create('Errore!');
      Exit;
    end;
    lv.Clear;
    XMLDoc.FileName := File_Locale;
    XMLDoc.Active:=True;
    StartItemNode:=XMLDoc.DocumentElement
                   .ChildNodes.First.ChildNodes.FindNode('item');
    ANode := StartItemNode;
    repeat
      STitle := ANode.ChildNodes['titolo'].Text;
      sDesc := ANode.ChildNodes['descrizione'].Text;
      with LV.Items.Add do
      begin
        Caption := STitle;
        SubItems.Add(sDesc);
      end;

      ANode := ANode.NextSibling;
    until ANode = nil;
  finally
    DeleteFile(File_Locale);
    Screen.Cursor:=crDefault;
  end;
```

```
end;
end.
```

COME INTERAGIRE CON UN DATABASE ACCESS

Java realizza le connessioni ai database attraverso il componente noto come JDBC. Poiché Java è multipiattaforma, tale insieme di API fornisce funzionalità generiche per l'accesso alle fonti condivise, indipendentemente dal loro formato. Il trucco è abbastanza semplice: JDBC fa da "tramite" tra un'applicazione Java ed il DBMS che si intende sfruttare, spesso attraversando anche altri collegamenti interni.

Tip fornito dal sig. C. Pelliccia

Per testare la connessione con gli archivi di Access, realizza un archivio chiamato miodatabase.mdb, e al suo interno crea una tabella nominata Persone, suddivisa in quattro campi:

- **ID,** un banale contatore usato come chiave primaria.
- **Nome,** di tipo testuale.
- **Cognome,** di tipo testuale.
- **Indirizzo,** di tipo testuale.

Quindi popola la tabella con qualche dato arbitrario.
A questo punto è necessario far riconoscere al sistema operativo il database, in modo che possa essere introdotto tra le fonti di dati ODBC gestite. Per far ciò, è necessario configurare un DSN.
L'operazione è semplice:

1. Dal *"pannello di controllo"* di Windows accedi alla voce *"origine dati ODBC"*.

2. Seleziona *"aggiungi"* nella scheda *"DSN utente"*.

3. Scegli, dall'elenco presentato, il driver di Access, identificato solitamente dalla dicitura *"Microsoft Access Driver (*.mdb)"*.

4. Nella maschera ora presentata, immetti i dati necessari per la creazione del DSN. Usa il nome *MioDatabase*, inserisci una descrizione a piacere, e con il pulsante *"seleziona"* collega il DSN al file *miodatabase.mdb* precedentemente creato.

5. Conferma ogni operazione effettuata, salvando così il DSN utente creato.

Ora la base di dati è ufficialmente riconosciuta dal sistema operativo, e JDBC può accedervi sfruttando, come tramite, il driver ODBC del sistema. In casi come questo, infatti, si parla di ponte JDBC-ODBC. Le classi Java che permettono l'accesso ai database sono contenute nel package *java.sql*.
Ecco un esempio in grado di sfruttare il database di Access appena registrato nel sistema:

```
import java.sql.*;

class JavaDatabase {

 public static void main(String[] args) {
  try {
   // Carico il driver JDBC-ODBC
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   // Apro una connessione con il database
   Connection con = DriverManager.getConnection(
    "jdbc:odbc:MioDatabase"
   );
   // Preparo l'oggetto che mi permetterà
   // l'utilizzo di SQL
   Statement stat = con.createStatement();
   // Eseguo l'interrogazione, ottentendo
   // un oggetto ResultSet come risultato
   ResultSet res = stat.executeQuery(
    "SELECT * FROM Persone"
   );
   // Scorro ogni record presente e stampo
   // l'output sul video
   while (res.next()) {
    System.out.println(res.getString("Nome") + ", " +
                          res.getString("Cognome") + ", " +
                               res.getString("Indirizzo"));
   }
   // Chiudo il ResultSet
   res.close();
   // Chiudo lo Statement
   stat.close();
   // Chiudo la connessione
   con.close();
  } catch (Exception e) {
   System.out.println("Problema: " + e.toString());
  }
 }

}
```

Per prima cosa viene caricato il driver JDBC-ODBC:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Quindi la classe *Connection* è sfruttata per aprire la connessione verso il database. La stringa utilizzata, ovviamente, non è casuale:

```
jdbc:odbc:MioDatabase
```

MioDatabase, infatti, è proprio il nome del DSN creato pocanzi. A questo punto è superfluo commentare il resto:

il package *java.sql* contiene tutte le funzionalità necessarie per ogni scopo, e la documentazione ufficiale *(http://java.sun.com/docs/)* le illustra nel dettaglio.

DOWNLOADARE UN FILE DAL WEB E SALVARLO IN LOCALE

La procedura proposta consente di scaricare un file partendo dal suo URL e di salvare lo stesso in una directory locale. Affinché il tip possa funzionare è necessario importare i package *java.net* e *java.io*.

```
public static void download(String url, String filename)
{
  try
  {
    URL u = new URL(url);
    URLConnection conn = u.openConnection();
    InputStream in = conn.getInputStream();
    FileOutputStream out = new FileOutputStream(filename);
    byte[] bytes = new byte[1024];
    int l = 0;
    while ((l = in.read(bytes)) != -1)
    out.write(bytes,0,l);
    out.close();
    in.close();
  }
  catch (Exception e)
  {
    System.out.println("Problema: " + e.toString());
  }
}
```

La seguente riga di codice:

```
download("http://www.ioprogrammo.it/hello.zip",
                                    "hello_ioprogrammo.zip");
```

recupera il file denominato *hello.zip*, presente all'URL indicato, e lo salva sul proprio hard-disk con il nome *hello_ioprogrammo.zip*

UNA FUNZIONE TRIM EVOLUTA...

Il tip proposto emula la funzione *trim* di Visual Basic, ovvero rimuove gli eventuali spazi presenti all'inizio o alla fine di una stringa; utile da utilizzare anche solo lato client, nel browser.

Tip fornito dal sig. C. Miele

```
<%@ LANGUAGE="JScript"%>
function trim(stringa)
{
  var i;
  var inizio=-1;
  var fine=stringa.length;
        for(i=0;i<stringa.length;i++)
        {
                if(stringa.charAt(i)==" ")
                   inizio=i;
                else
                     break;
        }
        for(i=stringa.length-1;i>0;i--)
        {
              if(stringa.charAt(i)==" ")
                 fine=i;
              else
              break;
        }
        stringa = stringa.substring(inizio+1,fine);
        return(stringa);
}
function ltrim(stringa)
{
  var i;
  var inizio=-1;
     for(i=0;i<stringa.length;i++)
     {
        if(stringa.charAt(i)==" ")
           inizio=i;
        else
              break;
        }
         stringa = stringa.substring(inizio+1,stringa.length);
        return(stringa);
  }
function rtrim(stringa)
{
  var i;
  var fine=stringa.length;
  for(i=stringa.length-1;i>0;i--)
{
   if(stringa.charAt(i)==" ")
     fine=i;
   else
    break;
 }
  stringa = stringa.substring(0,fine);
  return(stringa);
}
%>
```

PHP

LA VERIFICA DELLA CARTA DI CREDITO

Questa funzione sviluppata in linguaggio PHP verifica l'esattezza di un codice di partita iva. La funzione restituisce 1 nel caso in cui il codice risulta corretto. Per l'utilizzo basta includere il programma in un file e richiamare la funzione *CheckIVA* passando come argomento la partita iva da controllare

Tip fornito dal Sig. R. Sensale

```php
<?php
## Creato da Rosario Sensale  ##
## E.Mail gsensa@freemail.it  ##
function CheckPIva($partita_iva)
{
  $dimensione=strlen($partita_iva);
  $somma_disp=0;
  $somma_pari=0;
  $somma_totale=0;
  $temp=0; //ritorna 0 se c'è un errore e 1 nel caso contrario
  //controllo la dimensione
  if ($dimensione == 11)
  {
      $somma_disp=intval($partita_iva[0]) + intval($partita_iva[2])
                + intval($partita_iva[4]) + intval($partita_iva[6]) +
                                                intval($partita_iva[8]);
      //controllo pari
      $temp_somma=intval($partita_iva[1])*2;
      $temp_str=strval($temp_somma);
      if ($temp_somma >= 10){
          $somma_pari=$somma_pari + intval($temp_str[0]) +
                                                intval($temp_str[1]);
      }
      else
      {$somma_pari=$somma_pari + $temp_somma; }
      $temp_somma=intval($partita_iva[3])*2;
      $temp_str=strval($temp_somma);
      if ($temp_somma >= 10){
          $somma_pari=$somma_pari + intval($temp_str[0]) +
                                                intval($temp_str[1]);
      }
      else
      {
          $somma_pari=$somma_pari + $temp_somma;
      }
      $temp_somma=intval($partita_iva[5])*2;
      $temp_str=strval($temp_somma);
      if ($temp_somma >= 10){
          $somma_pari=$somma_pari + intval($temp_str[0])
                                              + intval($temp_str[1]);
      }
      else
      {
          $somma_pari=$somma_pari + $temp_somma;
      }
      $temp_somma=intval($partita_iva[7])*2;
      $temp_str=strval($temp_somma);
      if ($temp_somma >= 10){
          $somma_pari=$somma_pari + intval($temp_str[0])
                                              + intval($temp_str[1]);
      }
      else
      {
          $somma_pari=$somma_pari + $temp_somma;
      }
      $temp_somma=intval($partita_iva[9])*2;
      $temp_str=strval($temp_somma);
      if ($temp_somma >= 10)
      {
          $somma_pari=$somma_pari + intval($temp_str[0]) +
                                                intval($temp_str[1]);
      }
      else
      {
          $somma_pari=$somma_pari + $temp_somma;
      }
      //somma totale
      $somma_totale=$somma_pari+$somma_disp;
      //elemento unita'
      $temp_str=strval($somma_totale);
      $controllo=10-intval($temp_str[1]);
      if (intval($partita_iva[10]) == $controllo)
      {
          $temp=1;
      }
      else
      { $temp=0;}
}
else
{ $temp=0; //non sono 11 caratteri}
return $temp;
}
?>
```

Robotica: hardware e software

Braccio meccanico: rotazione del polso

Al termine della lettura di queste pagine avremo visto come gestire la rotazione del polso della nostra mano meccanica, al fine di rendere possibile al nostro Robot lo svolgimento di compiti complessi.

L'esecuzione di compiti complessi da parte di un Robot, che vadano oltre il semplice spostamento di oggetti, presuppone la capacità di poter ruotare quell'appendice che tecnicamente viene chiamata manipolatore, ma che a noi piace chiamare 'mano'. Nello svolgimento delle più comuni azioni della vita quotidiana, diamo per scontato di conoscere le azioni meccaniche che ci consentono di compiere anche i gesti più semplici.

LA SCELTA DEGLI ATTUATORI

Per azionare le articolazioni del braccio meccanico, abbiamo bisogno di attuatori leggeri, semplici da controllare, potenti ed affidabili. I servocomandi 'Pulse Width Modulation' (PWM), sono un tipo particolare di attuatore elettromeccanico, comunemente utilizzato in applicazioni di modellismo dinamico, che basano il loro funzionamento sulla decodifica di un treno di impulsi che viene inviati sul terminale di controllo. In questa sede faremo riferimento a quel tipo di attuatori dotati di blocco, che ne limita l'escursione a circa +/-100° rispetto alla posizione centrale, anche se esistono servomeccanismi di questo genere liberi di ruotare sul loro asse, come semplici motori. Le caratteristiche meccaniche di questi servocomandi li rendono molto interessanti: la coppia utile risulta essere di almeno 2,5 Kg cm per i modelli più leggeri, fino a raggiungere un valore di oltre 15 Kg cm. Esternamente, il servocomando è molto semplice e compatto e si adatta perfettamente alle applicazioni di Robotica. Il congegno presenta tre terminali, due dei quali (rosso e nero), servono alla sua alimentazione, mentre il terzo (bianco) permette l'invio del treno di impulsi di comando. In commercio esistono diversi modelli di servocomando, ma tutti riconoscono lo stesso standard di controllo. Il parametro fondamentale è la durata dell'impulso positivo di controllo che può variare tra 1 mSec e 2 mSec, valori che corrispondono alle due posizioni estreme. Le due figure riportate in questa sede mostrano il movimento rotatorio del servocomando, al variare della lunghezza dell'impulso di controllo. La frequenza degli impulsi, entro certi limiti non è un parametro critico, generalmente questo valore si aggira intorno ai 20-30 HZ. Ovviamente, se si desidera posizionare il servoco-

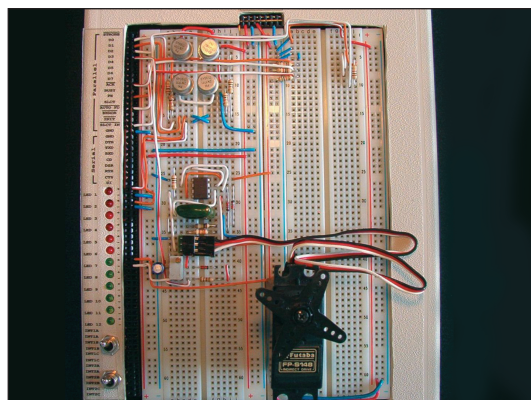



Fig. 1: Il servocomando utilizzato per la rotazione del polso del Robot è reperibile in qualunque negozio di hobbistica e modellismo.

Fig. 2: Come riportato in figura per posizionare il servocomando a fondo corsa in senso antiorario, rispetto alla direzione della mano, occorre inviare un treno di impulsi lunghi 2 mSec.

☑ QUESITI ALL'AUTORE
L'autore è lieto di rispondere ai quesiti dei lettori sull'interfacciamento dei PC all'indirizzo: luca.spuntoni@ioprogrammo.it

mando nella posizione centrale, sarà sufficiente inviare un treno di impulsi positivi, della durata di 1,5 Msec. Le tolleranze costruttive dei componenti, servocomando incluso, potrebbero comportare una deviazione della posizione reale, rispetto a quella richiesta, a questo scopo è stato inserito nello schema elettrico, un potenziometro per effettuare la regolazione dell'escursione dell'attuatore. Completata la descrizione funzionale degli attuatori, procediamo ad analizzarne le metodologie di implementazione, prima nella struttura hardware e poi nel programma di gestione software.

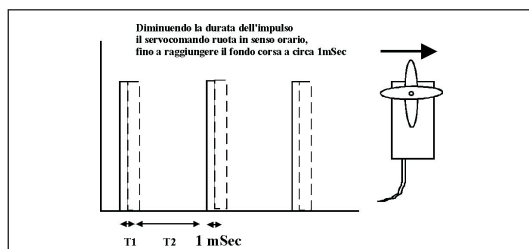



Fig. 3: ***Diminuendo la durata degli impulsi, l'attuatore ruota progressivamente fino a raggiungere l'altro estremo quando gli impulsi hanno una lunghezza di 1 mSec.***

L'IMPLEMENTAZIONE DEL CONTROLLO

Per completezza, si riporta di seguito lo schema a blocchi dei circuiti di controllo del braccio meccanico realizzato fino a questo punto. Nella tabella riportata di seguito, si riassumono tutte le caratteristiche salienti del sistema, che verranno poi tradotte sotto forma di schema elettrico più avanti in queste pagine. Per gestire il movimento dell'attuatore, utilizziamo il bit *D3* della porta dati, presente sul retro del nostro PC, nella porta parallela. Attraverso questa linea provvederemo ad inviare un segnale di comando del circuito di gestione che analizzeremo più avanti. In questo modo vedremo come sia possibile comandare il movimento di un attuatore a modulazione della lunghezza dell'impulso (PWM), controllando l'apposito circuito elettronico. Il lettore potrà risalire alle funzionalità delle altre linee consultando gli articoli dello stesso autore, pubblicati nei due numeri precedenti.

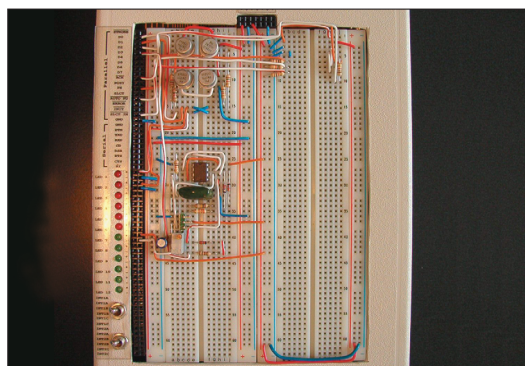



Fig. 4: ***Il circuito viene realizzato senza la necessità di effettuare saldature, per mezzo della apparecchiatura 'PC Explorer light'.***

LO SCHEMA ELETTRICO

La progettazione di un circuito che svolga il controllo di un servomeccanismo PWM non è complicata, ma presuppone una serie di conoscenze elettroniche il cui insegnamento esula dallo scopo di queste pagine. Volutamente si cercherà di evitare una terminologia troppo specializzata, rimanendo a disposizione per i lettori che vogliano approfondire l'argomento in forma epistolare, attraverso l'indirizzo di posta elettronica che viene riportato a lato. E' stato descritto il principio di funzionamento dei servomeccanismi PWM *(Pulse Width Modulation)*: il modo attraverso il quale si vuole generare il treno di impulsi, in merito al quale si è già discusso, è quello di realizzare un circuito oscillatore per mezzo dell'integrato *555*. Senza scendere in ulteriori dettagli, diciamo che la lunghezza dell'impulso positivo viene stabilito dai valori dei componenti *R1, R2* e *C1*, che determinano la costante di tempo di carica del con-

PIN PC Master	SEGNALE	DIREZIONE	TIPO PORTA	PIN PORTA	DESCRIZIONE
2	D0	OUT	PARALLELA DATA D0	2	Open Hand
3	D1	OUT	PARALLELA DATA D1	3	Close Hand
4	D2	OUT	PARALLELA DATA D2	4	INFRA RED sensor EMITTER
5	D3	OUT	PARALLELA DATA D3	5	PWM Wrist rotation
10	$\overline{\text{ACK}}$	IN	PARALLELA STATUS S6	10	INFRA RED sensor RECEIVER
11	BUSY	IN	PARALLELA STATUS /S7	11	Reserved
12	PE	IN	PARALLELA STATUS S5	12	Pression sensor 1
17	SLCT IN	IN	PARALLELA STATUS S4	17	Pression sensor 2
14	$\overline{\text{AUTO FD}}$	IN	PARALLELA STATUS S3	14	Pression sensor 3
18	GND	PARALLEL GND	PARALLELA	18-25	Signal Ground

Tab. 1: Tabella riassuntiva del sistema.

☑ SCHEMI A BLOCCHI

Su richiesta dei lettori, gli schemi a blocchi sono stati inseriti nel CD-Rom all'interno del file:

'ROBOT_Mano_e_Rotazione_del_Polso.ZIP'

☑ IL BRACCIO MECCANICO E PC EXPLORER LIGHT

Per maggiori informazioni sul braccio meccanico, sulle interfacce relative e sull'apparecchiatura *'PC Explorer light'* è possibile visitare il sito:

http://web.tiscali.it/spuntosoft/

densatore *C1* (ed il tempo in cui raggiunge un certo valore di soglia corrispondente a *T1*), mentre la sua scarica, proporzionale all'intervallo di tempo *T2* dipende soltanto da *R3* e *C1*. Per i componenti scelti nel nostro caso, possiamo calcolare i valori degli intervalli di tempo *T1* ,*T2*, nonché della frequenza di oscillazione *F* e del *'Duty Cycle' D*, corrispondente percentualmente a *T1/(T1+T2) *100,* che in poche parole fornisce un'idea immediata del tempo in cui la linea di uscita è a livello logico *'1'*. In Tab. 2 sono mostrati i valori di progetto, nel caso in cui sul piedino *N 5* dell'integrato si trovino i *2/3* della tensione di alimentazione. Il lettore attento noterà che la frequenza di uscita è di 62 HZ, rispetto ai 20-30 HZ considerati come riferimento: questo è dovuto al fatto che abbiamo inserito una resistenza variabile *(R4)* che ha lo scopo di regolare il punto di 'zero' del servocomando aggiustando la durata dell'impulso *T1,* variando la tensione di 'commutazione' dei comparatori di Soglia e Trigger presenti all'interno dell'integrato. Variando la tensione ai capi del piedino *N5* possiamo quindi variare la posizione del servocomando: quando applichiamo +4,81V questo si posizionerà a fondo corsa in senso orario, mentre fornendo 1,84 V a fondo corsa in senso antiorario: un valore intermedio farà posizionare l'attuatore in una posizione intermedia in modo proporzionale. Commutando il Bit *D3* della nostra porta parallela, varieremo la tensione sul piedino *N5* attraverso *D3* ed *R4* tra questi estremi di tensione, facendo ruotare il polso del robot a fondo corsa da un lato e dall'altro. Per ottenere un posizionamento più accurato dovremo aggiungere un ulteriore circuito di conversione digitale-analogico, che tratteremo nel prossimo appuntamento. Sul lato destro dello schema si possono notare le connessioni alle linee della porta parallela, o dell'apparecchiatura *'PC Explorer light',* sulla quale è possibile avere maggiori informazioni visitando il sito: *http://web.tiscali.it/spuntosoft/*, oppure scrivendo all'indirizzo: *spuntosoft@tiscali.it*. L'architettura del sistema rende possibile il massimo controllo software del braccio meccanico, rendendo disponibili tutti i segnali di uscita ed ingresso per la sua gestione: il circuito elettrico è volutamente molto semplice, occorre però un accurato controllo degli errori e delle condizioni logiche proibite attraverso il software di gestione.

☑ IL BRACCIO MECCANICO IN AZIONE

Nel CD allegato alla rivista sono disponibili due filmati che mostrano la mano meccanica in azione.

Robot_rotazione_del_polso1.AVI

Robot_rotazione_del_polso2.AVI

LA REALIZZAZIONE DEL CIRCUITO

La lista dei componenti necessari viene riportata a lato di queste pagine e nel circuito elettrico, per comodità e su richiesta dei lettori all'interno del file: *ROBOT_Mano_e_Rotazione_del_Polso.ZIP*, incluso nel CD ROM allegato alla rivista, con il nome: *Rotazione_del_polso_schema_Elettrico.bmp*. Provvediamo ad inserire prima i componenti più piccoli ed a profilo più basso, ovvero il circuito integrato, i diodi e le resistenze, posizioniamo quindi il potenziometro ed i condensatori. Provvediamo ad eseguire i collegamenti per mezzo di spezzoni di filo rigido, cercando di eseguire un cablaggio ordinato, per facilitare l'eventuale ricerca degli errori. Il montaggio della parte relativa alla gestione del motore della mano meccanica e della parte relativa ai sensori di pressione e prossimità sono stati pubblicati sui numeri di ioProgrammo dei due mesi precedenti. Il cablaggio può essere eseguito facilmente utilizzando l'apparecchiatura mostrata in figura, chiamata 'PC Explorer light', la più semplice della famiglia 'PC Explorer', sulla quale è possibile avere maggiori informazioni sul sito '*http://web.tiscali.it /spuntosoft/*', oppure scrivendo all'indirizzo *spuntosoft@tiscali.it*. In alternativa è possibile utilizzare le tecniche costruttive convenzionali, ovvero dotandosi di stagno, saldatore ed una buona dose di pazienza: il lettore ha in ogni caso tutte le informazioni necessarie alla realizzazione della parte hardware e più avanti troverà il software di gestione, completo di componenti pronti all'uso, del programma compilato e funzionante dotato di codice sorgente.

		uF	pF
C1	2,2E-07	0,22	220000
R1	1000		
R2	3300	**T1**	**T2**
R3	100000	0,000656	0,015246
F	**62,8868**		
D	**4,12272**		

Tab. 2: Valori del progetto con N5=2/3V.

IL SOFTWARE DI CONTROLLO

Il software di controllo è il responsabile della gestione di tutta la applicazione: gli schemi elettrici infatti

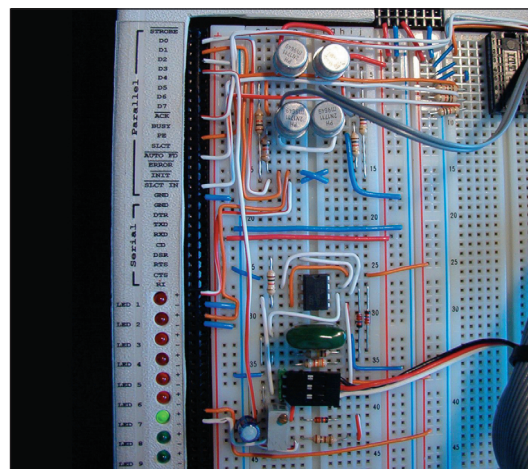



***Fig. 5:** L'immagine mostra il circuito completamente montato.*

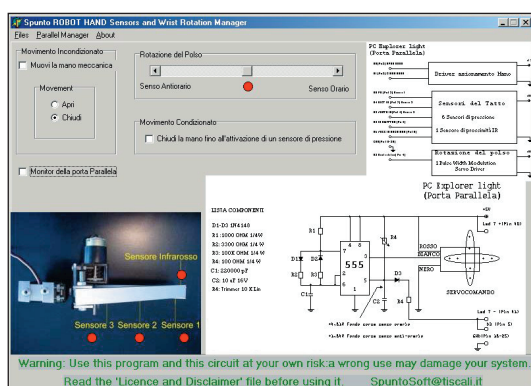



Fig. 6: *Il programma è dotato di tutti i comandi relativi al movimento di apertura della mano meccanica, del controllo dei sensori di pressione e di prossimità IR e della rotazione del polso.*

sono ridotti all'essenziale e deve essere posta la massima cura da parte del programmatore affinché condizioni proibite nello stato logico delle linee hardware di controllo non vengano a verificarsi. Il programma Delphi descritto di seguito ha la caratteristica di accedere all'hardware del PC attraverso i propri indirizzi fisici di I/O. Questa tecnica, dal momento che scavalca il sistema operativo, potrebbe non 'piacere' a Windows NT, 2000, oppure XP, pertanto si consiglia di utilizzare un calcolatore dotato di Win 3.X, Win 9X, oppure Millennium. In alternativa occorre scrivere una parte di codice che gestisca i privilegi del sistema, per non incorrere ad un errore del tipo *Privileged error.* Una ulteriore alternativa che risolve ogni problema è la scrittura di un appropriato 'Device Driver', che però esula dallo scopo di queste pagine, data la complessità dell'argomento. La classe principale, il codice della quale si trova sul CD allegato alla rivista, ingloba tutte le funzionalità del programma, in particolare per quanto riguarda la gestione del servocomando notiamo le procedure *SetWristPWMLow* e *SetWristPWMHigh*, che si occupano rispettivamente di posizionare il pin *N5* dell'integrato *555* a livello basso ed alto. La variabile globale *PWMCounter* rappresenta il valore di un loop chiuso, che ha lo scopo di generare un segnale di *Duty Cycle* variabile, in funzione della posizione della scrollbar *ScrollBarWrist,* come viene meglio specificato di seguito. Alla creazione della finestra principale, viene eseguita la procedura *FormCreate,* nella quale vengono inizializzati tutti i parametri fondamentali per la corretta esecuzione del programma. In particolare è importante notare l'inizializzazione delle variabili: *ParallelPortDataAddress* e *ParallelPortStatusAddress,* impostate per default sui valori di *LPT1,* per cui se il lettore ha intenzione di utilizzare una porta diversa dovrà impostare queste variabili in modo congruente con gli indirizzi fisici del suo sistema. La procedura *HandAction* è il cuore della gestione dei sensori della mano, del movimento della pinza e della rotazione del polso. In merito ai primi due aspetti, si rimanda il lettore alla lettura dei due articoli dello stesso autore, pubblicati nei due numeri precedenti di questa rivista. Per quanto riguarda la rotazione del polso, diciamo che la procedura *HandAction* viene chiamata da un apposito timer ogni 50 millisecondi, viene generato quindi un segnale ad onda rettangolare, con un duty cycle variabile tra 0 e 100%, in funzione della posizione dello scrollbar *ScrollBarWrist,* chiamando le procedure *SetWristPWMHigh* e *SetWristPWMLow*: l'interpretazione di questo segmento di codice è autoesplicativo.

```
procedure TSpuntoROBOTHandWristForm.HandAction;
                                   // Controls the ROBOT's Hand
Var
   W,DataPortValue:Word;
Begin
  // Wrist Control
  Inc (PWMCounter);                //Loops from 1 to 20
  If PWMCounter>20 then PWMCounter:=1;
                                          //Every 50 mSec
  If ScrollbarWrist.Position>=PWMCounter then begin
     SetWristPWMHigh;
  End
  Else Begin
     SetWristPWMLow;
  End;
   // Unconditioned Hand Movement
   If MoveHandCheckBox.Checked then begin
      CloseHandUntilAnyPressionSensor.Checked:=False;
   If CloseRadioButton.Checked then CloseHand
                                        else OpenHand;
   End
   Else Begin
   If SpuntoROBOTHandWristForm.Visible then
      StopHand;
   End;
   // Reads Hand Sensor Status and updates the LEDs
   ReadHandSensorsStatus;
   // Conditioned Hand Movement
   If CloseHandUntilAnyPressionSensor.Checked then begin
      MoveHandCheckBox.Checked:=False;
       // Start closing the hand
       CloseHand;
       //Waits the pression sensor activation
       IF IsAnyPressionSensorPressed Then Begin
          CloseHandUntilAnyPressionSensor.Checked:=False;
          // Stops the hand
          StopHand;
       End;
    End;
End;
```

Le procedure *SetWristPWMHigh* e *SetWristPWMLow,* riportate di seguito, provvedono a leggere il valore della porta, a posizionare il bit *D3* allo stato logico *1*, oppure *0* rispettivamente ed ad impostare il LED presente sulla form in modo appropriato.

☑ COMPONENTI NECESSARI

N1 Servocomando PWM

N1 Circuito integrato 555
N3 Diodi 1N4148
N1 Resistenza 1 K Ohm ? W
N1 Resistenza 3,3 K Ohm ? W
N1 Resistenza 100K Ohm ? W
N1 Resistenza 100 Ohm ? W
N1 Condensatore 220000 pf poliestere
N1 Condensatore 10 uF elettrolitico 16 V
N1 Trimmer 10 K Ohm lineare

```
procedure TSpuntoROBOTHandWristForm.SetWristPWMLow;
                                        // Sets Wrist PWM to Low
Var
W,DataPortValue:Word;
Begin
   DataPortValue:=Readport(ParallelPortDataAddress);
                                        // Reads Port Data
   W:=(DataPortValue and $F7);     // Sets bit D3 to '0'
   WritePort(ParallelPortDataAddress,W);
   SpuntoLedWrist.LedOn;
End;
procedure TSpuntoROBOTHandWristForm.SetWristPWMHigh;
                                        // Sets Wrist PWM to High
Var
W,DataPortValue:Word;
Begin
   DataPortValue:=Readport(ParallelPortDataAddress);
                                        // Reads Port Data
   W:=(DataPortValue OR $08);      // Sets bit D3 to '1'
   WritePort(ParallelPortDataAddress,W);
   SpuntoLedWrist.LedOff;
End;
```

Fig. 7: *Il software è dotato di un monitor della porta parallela, che permette l'accesso diretto ai tre indirizzi fisici delle porte Dati, di Status e di Controllo.*

Il programma discusso in precedenza è contenuto nel CD allegato alla rivista, completo di codice sorgente e file eseguibile, per motivi di brevità ne sono state analizzate solamente le parti fondamentali alla comprensione della gestione software della rotazione del polso della mano meccanica: per quanto riguarda il controllo del motore di azionamento della pinza e la gestione dei sensori, si invita il lettore a consultare gli articoli relativi pubblicati nei numeri precedenti.

COLLAUDO DEL SISTEMA

Prima di collegare il circuito al nostro PC occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato connesso come previsto: controlliamo che i connettori siano ben serrati e che nessuna parte metallica della mano possa urtare il circuito elettrico. Colleghiamo al nostro circuito il cavo relativo alla porta parallela del PC, se possediamo *PC Explorer* oppure *PC Explorer light* oppure provvedendo a costruire un cavo seguendo lo schema elettrico e la tabella riportati all'inizio dell'articolo. Alimentiamo il circuito e lanciamo il programma di gestione: la prima operazione da fare a questo punto è posizionare il cursore dello scrollbar in uno dei due estremi e verifichiamo che la mano meccanica ruoti a fine corsa dal lato corrispondente: provvediamo quindi a regolarne la posizione per mezzo del trimmer *R4*. Spostiamo il cursore dalla parte opposta e controlliamo che la mano ruoti di conseguenza all'altro estremo, se necessario ritorniamo a regolare il trimmer *R4*. Se il circuito non funziona, provvediamo a spegnere tutto prima di ricontrollare i collegamenti e riprovare di nuovo. La nostra applicazione di controllo della rotazione del polso della mano meccanica è a questo punto terminata: disponiamo di un sistema che è in grado di azionare la pinza, controllarne la presa per mezzo di sensori di pressione e verificare il corretto posizionamento dell'oggetto per mezzo del sensore a raggi infrarossi, nonché azionare la rotazione del polso del nostro braccio meccanico.

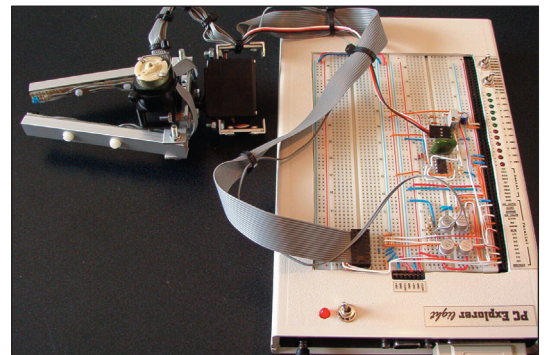



Fig. 8: *Dopo avere completato l'assemblaggio del circuito, siamo pronti a verificarne il funzionamento, collegando il sistema al PC per mezzo della porta parallela.*

CONCLUSIONI

Il progetto dello schema elettrico, tutti i collegamenti necessari, il software compilato ed i relativi codici sorgenti sono stati messi a completa disposizione del lettore: due filmati dimostrativi sono inoltre disponibili sul CD ROM allegato alla rivista.
Il lettore vorrà comprendere che nonostante quanto esposto in queste pagine sia stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo e di studio, pertanto l'editore e l'autore non sono da considerare responsabili per eventuali conseguenze derivanti dell'utilizzo di quanto esposto in questa sede, soprattutto per la tipologia e la complessità dell'argomento.

Luca Spuntoni

☑ STRUMENTAZIONE PER MONTAGGI SPERIMENTALI

Il sistema proposto in queste pagine è stato realizzato e collaudato con l'apparecchiatura per il collaudo e la sperimentazione di circuiti elettronici con Personal Computer 'PC EXPLORER light': ulteriori informazioni su come si possa reperire questa apparecchiatura è possibile visitare il WEB all'indirizzo:

http://web.tiscali.it/spuntosoft/

oppure inviare una e-mail a:

spuntosoft@tiscali.it.

JavaX: applicazioni multimediali in Java

Un player audio tutto Java

In questo articolo realizzeremo un lettore audio in linguaggio Java tramite l'impiego del package javax.sound.sampled, introdotto nella piattaforma Java 2, a partire dalla versione 1.3, e che consente una robusta e sofisticata gestione dell'audio digitale.

A partire dalla versione 1.3, SUN ha incluso nella piattaforma Java 2 le cosiddette Java Sound API: una collezione di classi ed interfacce che consentono la realizzazione di complessi e sofisticati sistemi di elaborazione dell'audio digitale. Le Java Sound Api sono state inserite in due differenti package: *javax.sound.midi* e *javax.sound.sampled*. Il primo package (del quale non ci occuperemo in quest'articolo) fornisce una serie di strumenti utili per la gestione di dati MIDI; il secondo, invece, mette a disposizione una ricca collezione di classi ed interfacce utili per la cattura, il mixaggio e la riproduzione dell'audio digitale. Scopo di questo articolo è quello di muovere i primi passi nello studio del package *javax.sound.sampled*, tramite la realizzazione di un semplice lettore audio sviluppato interamente in linguaggio Java.

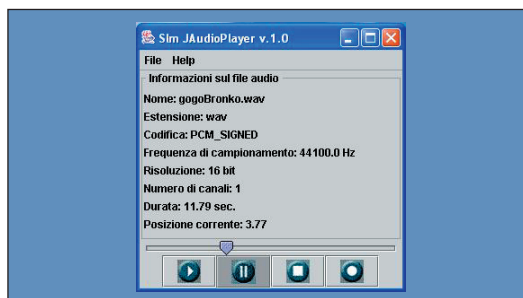
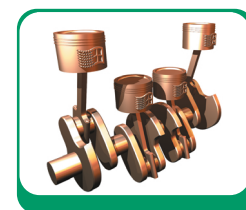



Fig. 1: Il lettore audio in azione.

CHE COSA OCCORRE PER REALIZZARE IL LETTORE AUDIO

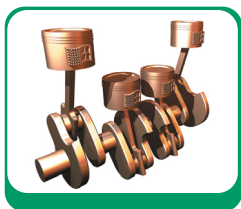
Per realizzare il nostro lettore audio avremo bisogno di tre elementi fondamentali: un sistema in grado di acquisire l'audio dall'esterno (da un file audio o direttamente da un microfono), un sistema in grado di riprodurlo all'esterno (ad esempio tramite degli altoparlanti) ed ovviamente un sistema di controllo in grado di coordinare tutte le operazioni di cattura, elaborazione e/o riproduzione del suono. In questo primo articolo ci limiteremo a realizzare un lettore audio in grado di aprire ed eseguire un file audio arbitrario (fra quelli supportati dal proprio sistema), mentre nel prossimo articolo verrà spiegato il modo in cui catturare e riprodurre l'audio proveniente direttamente da un microfono per poi salvarlo sotto forma di file audio. Nel CD allegato alla rivista è presente il codice sorgente dell'intera applicazione. Gran parte di esso è dedicato alla gestione dei componenti GUI, sulla quale però non ci soffermeremo, concentrandoci invece su quegli aspetti maggiormente legati alla gestione dell'audio digitale.

L'AUDIO DIGITALE ED IL PACKAGE JAVAX.SOUND.SAMPLED

Prima di addentrarci nella realizzazione del lettore di file audio, è bene spendere qualche parola sulla particolarità esclusiva del package *javax.sound.sampled*: la possibilità di gestire il segnale sonoro a livello di *"campione audio"*. Un campione audio è un "numero" che rappresenta l'ampiezza di un'onda sonora in un certo istante. E' noto infatti che ciò che il nostro cervello percepisce come "suono" è una successione di onde di pressione, dotate ciascuna di una certa intensità, che pervengono al nostro orecchio. I comuni microfoni convertono queste onde di pressione in tensioni elettriche proporzionali a tali intensità; un convertitore analogico-digitale (di cui sono dotate le attuali schede audio per PC) è in grado infine di convertire tali valori di tensione sotto

JAVA SOUND API HOME PAGE

All'indirizzo

http://java.sun.com/products/java-media/sound/index.html

è presente una sezione di java.sun.com interamente dedicata alle Java Sound Api, da cui è possibile scaricare, oltre alla documentazione ufficiale (e alla esaustiva Java Sound Programmer Guide), alcuni utili esempi dimostrativi.

forma di numeri, a loro volta proporzionali ai valori di tensione registrati. Ciascuno di tali numeri corrisponderà appunto ad un preciso campione audio. Il vantaggio principale dell'audio digitale è che, a questo punto, è possibile elaborare un segnale audio semplicemente agendo su tali numeri. Ad esempio, moltiplicare tutti i campioni audio per un fattore comune corrisponderà ad una variazione di scala del segnale acustico, in altre parole ad una variazione del suo volume. Una volta elaborato il segnale audio, è possibile infine riprodurlo acusticamente mediante un convertitore digitale-analogico (che converte i valori numerici dei singoli campioni in corrispondenti segnali di tensione) collegato a sua volta ad un altoparlante che è in grado di generare all'esterno nuove onde di pressione (il segnale acustico vero e proprio) in funzione dei segnali di tensione al suo ingresso. Ebbene, il package *javax.sound.sampled* mette a disposizione una serie di utili strumenti per effettuare, con grande semplicità, tutte le operazioni fin qui descritte. Ed ora all'opera!

☑ JAVA MEDIA FRAMEWORK

Per chi non ha la necessità di impiegare delle classi a così basso livello, come le Java Sound Api per la gestione dei file audio nelle proprie applicazioni, può risultare più conveniente ricorrere al Java Media Framework, un package che consente la gestione di numerosissimi formati multimediali (sia audio che video). Ulteriori informazioni sul Java Media Framework si possono trovare presso il sito http://java.sun.com/products/java-media/jmf/index.html

COME APRIRE UN FILE AUDIO

In breve, i file audio sono dei contenitori di campioni audio. In effetti, esistono molti modi con cui tali campioni possono essere immagazzinati in un file, ed è per questo che ogni file audio possiede generalmente una sezione iniziale (chiamata *header*) che specifica il cosiddetto formato del file: in altre parole, il modo in cui i campioni audio vengono in esso rappresentati (come ad esempio il numero di byte riservati per rappresentare ciascun campione).

Detto questo, la cosa da fare prima di tentare di aprire un file audio generico è accertarsi che il proprio sistema sia in grado di gestirlo. Attualmente, gli unici formati di file audio supportati da Java sono i formati *WAVE, AU, AIFF* e *SND* (la documentazione ufficiale della Sun non esclude però la possibilità che future implementazioni delle Java Sound Api supporteranno ulteriori formati audio). Tuttavia, non è affatto detto che ogni sistema sia in grado di gestire tutti i formati audio supportati da Java (il sistema dell'autore, ad esempio, è in grado di supportare i formati *WAVE, AIFF* ed *AU*, ma non il formato *SND*).

A questo scopo, le Java Sound Api ci mettono a disposizione la classe *AudioSystem*, che tra le altre cose è in grado di fornire un elenco completo dei dispositivi audio installati nel proprio sistema, nonché la lista di tutti i formati di file audio da esso supportati. In particolare, per quest'ultima operazione è possibile invocare il seguente metodo della classe *AudioSystem*:

```
AudioFileFormat.Type[] supportedAudioFormat =
                    AudioSystem.getAudioFileTypes();
```

che restituisce un array di oggetti di tipo *AudioFileFormat.Type*, una classe in grado di fornire informazioni utili per l'identificazione di uno specifico formato di file audio. Ad esempio, tale classe possiede il metodo *String getExtension()* che restituisce l'estensione del particolare formato di file audio supportato. In questo modo è possibile, tramite l'impiego di un oggetto *JFileChooser* e di un oggetto di tipo *ExampleFileFilter* (un'estensione della classe astratta *javax.swing.filechooser.FileFilter*), fornire all'utente una finestra di dialogo per la scelta del proprio file audio, grazie ad un particolare filtro che consente l'apertura dei soli file audio di estensione compatibile col sistema in uso:

```
JFileChooser chooser = new JFileChooser(currentPath);
ExampleFileFilter filter = new ExampleFileFilter();
   if (supportedAudioFormat==null) return false;
   int numFormats = supportedAudioFormat.length;
   if (numFormats<1) { System.out.println("Il sistema
     non supporta alcun formato audio" + "compatibile
                con l'attuale implementazione delle
                                  Java Sound Api");
     return false;}
   for (int i=0;i<numFormats;i++) {
     filter.addExtension(supportedAudioFormat[
                          i].getExtension());  }
    filter.setDescription("Tutti i file audio supportati");
    chooser.setFileFilter(filter);
    chooser.setDialogTitle("Apertura File audio");
    int returnVal = chooser.showOpenDialog(this);
    if(returnVal == JFileChooser.APPROVE_OPTION) {
      currentPath = chooser.getSelectedFile(
                          ).getAbsolutePath();
     return openAudioFile(new File(currentPath));
   }
```

A questo punto, ottenuto un riferimento valido ad un file audio supportato dal proprio sistema, è possibile procedere alla lettura del file vero e proprio. Le operazioni da effettuare sul file dovranno consistere nel:

leggere l'insieme dei campioni audio in esso contenuti (codificati tramite una sequenza di byte e rap-

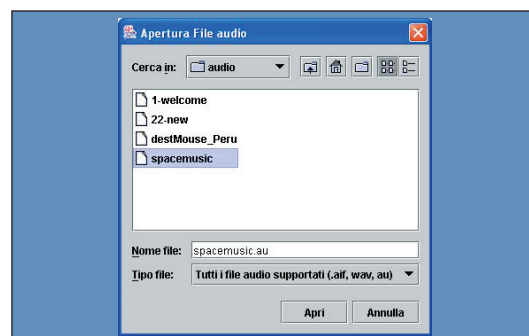



Fig. 2: *La finestra di dialogo per l'apertura del file audio.*

presentanti il segnale audio vero e proprio);
ottenere una serie d'informazioni relative al particolare formato dei campioni audio da leggere, come il numero di bytes impiegati per rappresentare ogni singolo campione, la frequenza di campionamento (ossia il numero di campioni audio necessari per rappresentare un secondo di segnale sonoro), il numero di canali impiegati (in altre parole, sapere se il segnale è mono oppure stereo), e così via.

Per la prima operazione è sufficiente invocare ancora una volta la classe *AudioSystem* per ottenere un riferimento ad un oggetto di tipo *AudioInputStream* (diretta sottoclasse di *java.io.InputStream*), che conterrà una sequenza ordinata di tutti i campioni audio presenti nel file. Per la seconda operazione, invece, è possibile ottenere un riferimento ad un oggetto di tipo *AudioFormat* direttamente dall'oggetto *AudioInputStream* appena creato. Quanto detto è realizzabile mediante le seguenti righe di codice:

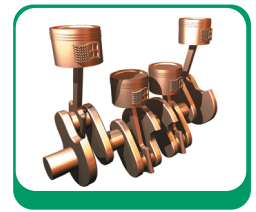
```
AudioInputStream ais = null;
AudioFormat af = null;
...
private boolean openAudioFile (File audioFile) {
  try{
      ais = AudioSystem.getAudioInputStream(audioFile);
      af = ais.getFormat();
  } catch (Exception ex){System.out.println(ex);
                                        return false; }
...
return true; }
```

Il metodo *AudioSystem.getAudioInputStream(audioFile)*, nel caso si tenti di aprire un file audio non supportato dal proprio sistema, scatena una eccezione apposita di tipo *UnsupportedAudioFileException.*

ASCOLTARE IL FILE AUDIO APERTO: L'IMPIEGO DELLE LINEE

Nelle Java Sound Api l'audio può essere catturato e/o riprodotto tramite le cosiddette "linee", particolari oggetti, che consentono ai campioni audio di interagire con i dispositivi audio di I/O del proprio sistema. Tali oggetti implementano l'interfaccia Line insieme alle sue derivate di tipo *DataLine*: l'interfaccia *TargetDataLine,* che consente l'acquisizione audio da un dispositivo di input come il microfono (e su cui si tornerà a parlare nel prossimo articolo), e le interfacce *SourceDataLine* e *Clip* che consentono, invece, l'invio dei campioni ad un dispositivo audio di output (come ad esempio un altoparlante). Per riprodurre musicalmente il file aperto, in effetti, potremmo impiegare indifferentemente sia un oggetto di tipo *SourceDataLine* che uno di tipo *Clip*. Tuttavia, mentre gli oggetti di tipo *SourceDataLine* sono stati ideati per supportare lo streaming di campioni audio in tempo reale (tramite il continuo aggiornamento, in fase di riproduzione audio, di in un apposito buffer interno contenente i prossimi campioni da suonare), gli oggetti *Clip* sono stati implementati per consentire il precaricamento in memoria di un intero brano audio (o comunque di una arbitraria porzione di esso, anche in relazione alle risorse di memoria del proprio sistema). Di conseguenza, risulta conveniente impiegare una linea di tipo *Clip* tutte le volte che è possibile disporre dell'intera sequenza dei campioni audio del brano da riprodurre (e di cui sia nota la dimensione); viceversa, è conveniente (se non necessario) impiegare una linea *SourceDataLine* ogniqualvolta non si può conoscere a priori la lunghezza del segnale audio da riprodurre (si pensi ad esempio al caso di un'applicazione che deve amplificare in tempo reale una voce catturata da un microfono, o ancora al caso in cui un file audio è troppo lungo per poter essere immagazzinato tutto in una volta in memoria).
D'altra parte, le linee di tipo *Clip* sono le uniche a fornire dei metodi diretti che consentono di riprodurre un numero arbitrario di volte il brano audio, nonché di specificare la posizione da dove far partire (o sul quale far terminare) la sua esecuzione. Nel nostro caso – avendo a che fare con dei campioni audio immagazzinati in un file audio (e potendo quindi risalire alla loro quantità) sarà conveniente impiegare un oggetto di tipo *Clip*. Le seguenti linee di codice mostrano il modo in cui è possibile ottenere una linea di tipo *Clip* a partire dall'oggetto *AudioInputStream ais* ricavato dal file audio aperto in precedenza:

Fig. 3: *La gerarchia delle linee nelle Java Sound Api.*

```
public Clip mySound = null;
...
private boolean setClipFromAudioInputStream() {
  DataLine.Info info = new DataLine.Info(Clip.class,
     ais.getFormat(), ((int) ais.getFrameLength() * ()));
    try{
       mySound = (Clip) AudioSystem.getLine(info);
       mySound.open(ais);
       ...
```

☑ LA CLASSE AUDIOSYSTEM DEL PACKAGE JAVAX.SOUND .SAMPLED

La classe *AudioSystem* è probabilmente la più importante di tutto il package *javax.sound.sampled*. I vari metodi messi a disposizione da tale classe consentono infatti, fra le altre cose, di ottenere un elenco completo di tutti i dispositivi audio installati nel proprio sistema, di indagare sui formati di file audio supportati, nonché di leggere e scrivere file audio di formato diverso.

```
    } catch (Exception ex) {System.out.println(ex);
return false;}
```

Per ottenere una linea di tipo *Clip* è necessario innanzitutto crearsi un oggetto di tipo *DataLine.Info*, il cui costruttore prende in ingresso, nell'ordine, i seguenti parametri:

un oggetto di tipo *Class* (che, a seconda del tipo di linea che si vuole creare, dovrà essere *SourceDataLine.class, TargetDataLine.class,* o *Clip.class*).
un oggetto di tipo *AudioFormat* (nel nostro caso specifico si tratta del formato dei campioni audio contenuti nel file precedentemente aperto che – si ricorda – sono stati memorizzati in un oggetto *AudioInputStream ais*)
la dimensione (espressa in numero di byte) del buffer in cui memorizzare i campioni audio. Nel caso di creazione di una linea di tipo *Clip*, esso corrisponderà al numero di byte necessari per contenere tutti i campioni audio del brano da riprodurre; nel caso invece si debba creare una linea di tipo *TargetDataLine* oppure *SourceDataLine,* tale valore dovrà indicare la dimensione del buffer interno utilizzato per la memorizzazione temporanea dei campioni audio rispettivamente da catturare o riprodurre di volta in volta. Sui criteri da adottare per una scelta appropriata della dimensione di tale buffer ritorneremo nel prossimo articolo.

Una volta creato il nostro oggetto *DataLine.Info info*, è finalmente possibile ottenere un riferimento ad una linea di tipo *Clip*, tramite la seguente istruzione:

```
mySound = (Clip) AudioSystem.getLine(info);
```

Tale operazione può scatenare una eccezione di tipo *LineUnavailableException,* qualora non sia possibile disporre del tipo di linea richiesta. Per poter rendere operativa la *Clip* appena creata, è necessario aprirla nel modo seguente:

```
mySound.open(ais);
```

in cui l'argomento *ais* è, ancora una volta, l'oggetto *AudioInputSream* contenente l'intera sequenza dei campioni audio del nostro file audio. In questo modo, il *Clip* può risalire alla dimensione da assegnare al proprio buffer interno per la memorizzazione di tutti i campioni audio presenti in *ais.* Nel caso il sistema non sia in grado di memorizzare l'intera sequenza audio (ad esempio perché il file audio è troppo lungo per poter essere immagazzinato tutto in una volta in un buffer interno) viene lanciata una eccezione di tipo *LineUnavailableException*. In quest'ultimo caso, per poter riprodurre il proprio file audio sarà necessario ricorrere ad una linea di tipo *SourceDataLine*, come vedremo nel prossimo articolo. Nel caso invece la linea di tipo *Clip* sia stata aperta senza problemi, da questo momento in poi sarà possibile impostare una posizione arbitraria all'interno del file a partire dalla quale far partire la riproduzione audio, mediante l'invocazione del metodo seguente:

```
mySound.setFramePosition(int frame)
```

dove il parametro *frame* specifica la posizione desiderata all'interno del file audio in termini di "numero di frammento musicale". Un *frame* musicale altro non è che un raggruppamento di byte necessari per rappresentare adeguatamente un istante di segnale sonoro. Così, ad esempio, nel caso il nostro file audio fosse dotato di una frequenza di campionamento di 8000 Hz (ovvero di 8000 campioni audio al secondo), e di una durata di 10 secondi, per poter ascoltare il brano dal quinto secondo in poi dovremmo impostare il seguente valore di frame: 8000*5 = 40000. Per far partire l'esecuzione musicale vera e propria, dalla posizione di *frame* corrente è sufficiente invocare il metodo *start():*

```
mySound.start();
```

mentre per metterla in pausa si deve ricorrere al metodo *stop():*

```
mySound.stop();
```

Se si desidera invece riprodurre l'effetto del pulsante di "stop" del player audio (che arresta l'esecuzione audio e la riavvolge al punto di partenza) è possibile operare semplicemente nel modo seguente:

```
/**
*  Metodo che fa cessare la riproduzione audio del brano
*  e lo riposiziona all'istante di esecuzione iniziale
*/
public void stopSound() { mySound.stop();
   mySound.setFramePosition(0); }
```

☑ LE INTERFACCE SOURCEDATALINE E CLIP

Le interfacce SourceDataLine e Clip consentono entrambe la riproduzione audio di un brano sonoro. La differenza sta nel fatto che, mentre una SourceDataLine può essere impiegata per lo streaming di campioni audio in tempo reale (come la riproduzione in tempo reale dell'audio catturato da un microfono), una Clip può essere impiegata solo in quei casi in cui è possibile disporre da subito dell'intera sequenza di campioni audio da riprodurre (come la riproduzione di tutti i campioni relativi ad un file audio).

RICAVARE LA DURATA IN SECONDI E LA POSIZIONE CORRENTE DEL FILE AUDIO

La durata (in secondi) del brano musicale caricato nell'oggetto *Clip* (alla luce di quanto spiegato nel paragrafo precedente) può essere calcolata nel modo seguente:

```
float duration = ((float) mySound.getFrameLength())/
                    mySound.getFormat().getSampleRate();
```

(dove il metodo *mySound.getFrameLength()* restituisce il numero di frame musicali appartenenti alla *Clip*), mentre la posizione corrente della *Clip* (sempre in secondi) può essere facilmente ricavata come segue:

```
/** Metodo che restituisce la posizione corrente dello
                                         stream audio
*  contenuto nella Clip corrente (in secondi)
*/
public float getCurTimePosition() {
 return (mySound==null) ?
 0.0f : (float) mySound.getMicrosecondPosition(
                                        )/1000000; }
```

RIPRODURRE CICLICAMENTE UNA PORZIONE DI FILE AUDIO

Un'altra caratteristica esclusiva delle linee di tipo Clip è quella di consentire la riproduzione ciclica di una porzione di sequenza musicale. Per far questo è necessario innanzitutto invocare il metodo *setLoopPoints* specificando il frame iniziale (*start_frame*) ed il frame finale (*end_frame*) da riprodurre:

```
mySound.setLoopPoints(int start_frame,int end_frame)
```

e quindi far partire l'esecuzione nel modo seguente:

```
mySound.loop(int count)
```

dove il parametro count indica il numero di volte in cui si desidera riprodurre la porzione di brano selezionata.

COME INDIVIDUARE LA FINE DELLA RIPRODUZIONE AUDIO

Ciascuna linea può essere dotata di un ascoltatore apposito di tipo *LineListener*, in grado di notificare la sua apertura (in seguito all'invocazione del metodo *open()* e la conseguente generazione di un evento di tipo *LineEvent.Type.OPEN*), la sua chiusura (in seguito all'invocazione del metodo *close()* e la conseguente generazione di un evento di tipo *LineEvent.Type.CLOSE*), l'istante iniziale di riproduzione (o cattura) dell'audio (in seguito all'invocazione del metodo *start()* e la conseguente generazione di un evento di tipo *LineEvent.Type.START*) e l'istante finale di riproduzione (o cattura) dell'audio (in seguito all'invocazione del metodo stop() e la conseguente generazione di un evento di tipo *LineEvent.Type .STOP*). In particolare, il raggiungimento della fine di un file audio equivale ad un evento di tipo *STOP*. Le seguenti righe di codice mostrano il modo in cui è possibile dotare la nostra *Clip* di un ascoltatore *LineListener*:

```
private LineListener lineListener = new LineHandler();
...
mySound.addLineListener(lineListener);
...
/**
 *  Classe che consente la gestione degli eventi relativi
 *  alle linee su cui viaggiano i campioni audio
 */
public class LineHandler implements LineListener {
  public void update (LineEvent ev) {
    if (ev.getType() == LineEvent.Type.STOP)
         { System.out.println("LineEvent: STOP ->
                       raggiunta la fine del file audio.");
           stopSound(); }
    }
}
```

Se necessario, è possibile rimuovere dalla linea il proprio ascoltatore, nel modo seguente:

```
mySound.removeLineListener(lineListener);
```

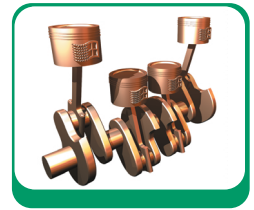
COME CHIUDERE I DISPOSITIVI AUDIO

Al termine dell'applicazione è buona cosa ricordarsi di chiudere tutti i dispositivi audio aperti in precedenza. Nel nostro caso, sarà sufficiente chiudere la nostra *Clip* nel modo seguente:

```
if (mySound!=null)
 { if (mySound.isOpen()) mySound.close();
    mySound = null;
 }
```

CONCLUSIONI

In questo articolo si è iniziato a parlare delle *Java Sound Api* mediante la realizzazione di un semplice lettore di file audio. In particolare, si è visto come individuare i formati di file audio compatibili col proprio sistema, come aprirli e come ascoltarli.
Nel prossimo articolo introdurremo nuove importanti funzionalità del nostro lettore audio, come la possibilità di catturare l'audio da microfono, riprodurlo ed infine salvarlo sotto forma di file audio, magari dopo avervi aggiunto qualche effetto sonoro. Non mancate all'appuntamento!

Stefano Monni

☑ SUL WEB

A partire dalla pagina all'indirizzo

http://java.sun.com/products/java-media/sound/

è disponibile la Java Sound Programmer Guide, concepita per tre categorie di lettori:

• SVILUPPATORI: programmatori che desiderano scrivere applicazioni o applet Java che fanno uso di audio o MIDI. La maggior parte dei lettori rientrano in questa categoria.

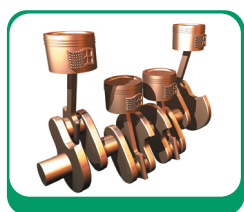
• FORNITORI DI SERVIZI: sviluppatori di moduli (plug-in) che estendono le capacità di implementazione della Java Sound Application Programming Interface (API). Ad esempio, un venditore potrebbe fornire un nuovo mixer audio o un sintetizzatore MIDI, o la capacità di leggere e scrivere un nuovo formato audio.

• API IMPLEMENTOR: sviluppatori che generano nuove implementazioni di Java Sound API.

Naturalmente, si presume che il lettore abbia una conoscenza di base di programmazione Java. La familiarità con l'audio e con il MIDI è utile ma non necessaria.

Interazione Telefono – Personal computer

"Scopri" chi ti telefona

L'identificazione del numero nelle chiamate in entrata, fino a poco tempo fa prerogativa delle compagnie telefoniche che offrono tale servizio, è ora possibile anche attraverso il modem del nostro PC.

Ciò è reso possibile dall'interfaccia *TAPI (Telephony Application Programmers Interface)*, progettata per permettere alle applicazioni l'impostazione ed il controllo dei dispositivi di telefonia. Attraverso questa interfaccia si possono avere informazioni sulle telefonate ricevute prima che si risponda, in particolare si possono ricevere informazioni circa il nome e il numero telefonico del chiamante.
Naturalmente sono richiesti sia l'abilitazione della linea telefonica a ricevere l'IdCaller ossia l'identificativo del chiamante, che la compatibilità del modem con l'interfaccia TAPI.

ACTIVEX

Il componente TAPI, utilizzato per la realizzazione dell'applicativo, è stato realizzato dalla Allen Martin inc. (*http://www.allen-martin-inc.com*). Tale componente viene distribuito in tre versioni: Lite, Professional e Enterprise. In Tab. 1 sono riportate sia le caratteristiche del componente, che le differenza tra le varie versioni. Per la realizzazione dell'applicativo *NumberID* (disponibile sul CD-Rom o sul Web) è stata utilizzata la versione Pro dell'ActiveX amTAPI.

☑ TAPI
Abbreviazione di Telephony API. Un´interfaccia telefonica standard per Microsoft Windows, progettata per permettere alle applicazioni l´impostazione e il controllo delle chiamate.

L'APPLICAZIONE NUMBER ID

Analizzando il Form del programma notiamo innanzitutto un controllo ComboBox *ComboLineName* che permette la visualizzazione e la scelta dell'apparato che interfaccerà la linea telefonica (*Device Line*). Selezionato il dispositivo, se ne potranno verificare sia le compatibilità (*CommandCaps*) che le impostazioni (*CommandSetting*).

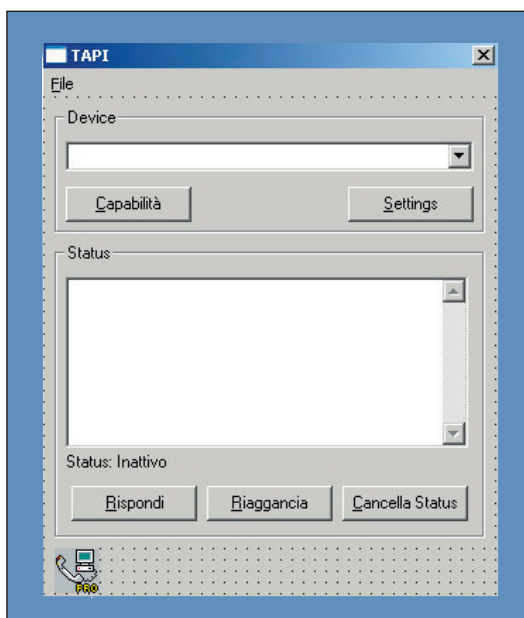



Fig. 1: *Il Form dell'applicazione.*

All'interno del Frame *Status*, oltre al controllo *TextCallState*, che ci fornisce tutto l'Output del programma, troviamo sia il pulsante *Rispondi* (*CommandAnswer*) che "aggancia" ad una telefonata in arrivo, sia il pulsante *Riaggancia* (*CommandHangUp*) che chiude la telefonata.

DEVICE LINE

Come evidenziato precedentemente, la prima

cosa da fare è identificare l'interfaccia con linea telefonica. Il componente *amTapi Pro* riesce a catturare, attraverso il codice che segue, tutti i *Device Line* disponibili sulla propria macchina:

```
For i = 0 To amTapiPro.NumberOfLines - 1
ComboLineName.AddItem amTapiPro.GetLineName(i)
    Next i
```

Così il *ComboLineName* risulterà popolato. Sarà quindi possibile effettuare una scelta dei vari dispositivi telefonici o modem rilevati dall'ActiveX.
La scelta nel nostro caso è ricaduta sul *Fax Modem Duxbury SM56 PCI.*
Per visualizzare le impostazioni del dispositivo scelto è necessario inserire il seguente codice:

```
Private Sub CommandSetting_Click()
        amTapiPro.ShowLineDialog Me.hWnd
End Sub
```

IL DATABASE TELEPHON BOOK

L'applicativo che abbiamo realizzato permette di conoscere, prima di rispondere alla telefonata in arrivo, il nome della persona che ci sta chiamando.
Questo è possibile poiché al programma è associato un Database Access che memorizza i nostri contatti telefonici.

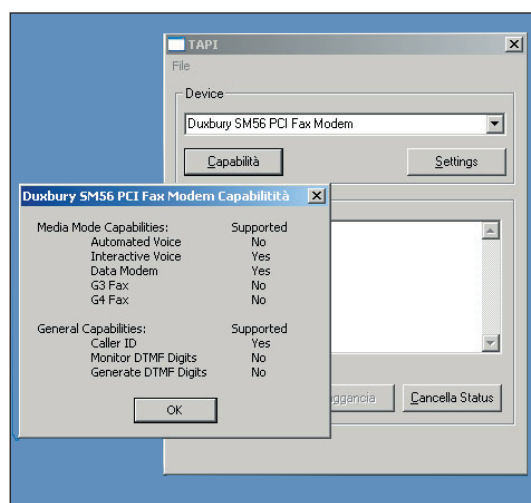
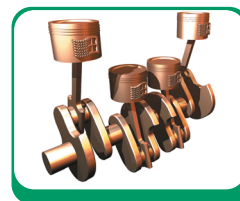



Fig. 2: ***Compatibilità del dispositivo telefonico.***

TelephonBook.mdb è un semplice DBMS con una sola tabella *"TelephonBook"*, che identifica il chiamante attraverso i campi: *Number, Name, Adress* e *Other Info*.

CHIAMATA IN ARRIVO

L'evento di chiamata in arrivo, *"Incoming-Call"*, è generato dal componente TAPI nel momento in cui si riceve il primo squillo telefonico.

Supported Features	Lite	Pro	Ent
Phone Lines Supported	Unlimited *2	Unlimited *2	Unlimited *2
Multiple addresses per line device			✓
Caller ID - Name, number and flags	✓	✓	✓
Called ID - Name, number and flags		✓	✓
SIT Tone Detection		✓	✓
Answer	✓	✓	✓
Hangup	✓	✓	✓
Call progress messages and call state	✓	✓	✓
Translate number - Using dialing properties	✓	✓	✓
Make Call - Dial full or partial number	✓	✓	✓
Dial - Completion of dialing on existing call	✓	✓	✓
Generate DTMF - Generate tones - 0-9, *, #, A-D	✓	✓	✓
Detect DTMF tones - remote party key press	✓	✓	✓
Call Privilege - Owner, Monitor, Owner + Monitor	✓	✓	✓
*3 Media Modes - Interactive Voice, Automated Voice, Data, Fax	✓	✓	✓
Data Rate Property For Data Modems		✓	✓
Location Dialog - Windows dialing properties dialog	✓	✓	✓
Enumerate Location Names		✓	✓
Get/set Current Location		✓	✓
Get Current Country Code - Useful to pre-fill number to dial		✓	✓
Get Current Area Code - Useful to pre-fill number to dial		✓	✓
Enumerate County Names		✓	✓
Line Dialog - Telephony device/modem setup dialog	✓	✓	✓
Get/Set Current Line Device	✓	✓	✓
Device Caps - Discover device hardware capabilities	✓	✓	✓
Enumerate line names - Installed telephony/modem devices	✓	✓	✓
Get/set current settings - data modem	✓	✓	✓
Line wave ID's (Record and Play wave files with amWave)	✓	✓	✓
Open Comm port handle - data modem (Use amComm for data stream)	✓	✓	✓
Negotiated Line TAPI version - OS, TSP, Application	✓	✓	✓
Negotiated Phone TAPI version - OS, TSP, Application		✓	✓
Speaker Phone Control - Microphone, Speaker, Mic/Speaker		✓	✓
Speaker Phone Volume and Gain Controls		✓	✓
Headset Control - Microphone, Speaker, Mic/Speaker		✓	✓
Headset Volume and Gain Controls		✓	✓
Handset Control - (Local Phone) Mic, Speaker, Mic/Speaker		✓	✓
Handset Volume and Gain Controls		✓	✓
Hook Switch Detection - On hook/Off hook		✓	✓
Phone wave ID's (Record and Play wave files with amWave)		✓	✓
Local Phone Ring		✓	✓
Phone Dialog - Telephony Phone device setup dialog		✓	✓
Local Phone Button Detection		✓	✓
Terminal Mode Support		✓	✓
Silence Detection		✓	✓
Pass Through - Direct communication with device hardware		✓	✓
Line Hold/Unhold		✓	✓
Line Blind Transfer		✓	✓
Dispose Metod -			
Dot Net Compatibilità		✓	✓
Call Handoff			✓
Transfer, park, conference calls and many other features to be announced. Feature requests and suggestions welcome.			✓

Tab. 1: Caratteristiche del componente della Allen Martin.

A questo punto *amTapi Pro* è in grado di contare gli squilli effettuati:

```
Private Sub amTapiPro_IncomingCall(ByVal
                                RingNumber As Long)
  LabelStatus.Caption = "Chiamata in arrivo"
  CallStateMessage "Squillo " & CStr(RingNumber)
End Sub
```

e di acquisire le informazioni relative al chiamante:

```
Private Sub amTapiPro_CallerID(ByVal Number As
   String, ByVal name As String, ByVal flags As Long)
  'Caller ID info
  If (flags And LINECALLPARTYID_NAME) > 0 Then
                      CallStateMessage "Nome: " & name
  If (flags And LINECALLPARTYID_NUMBER) > 0 Then
                  CallStateMessage "Numero: " & Number
  If (flags And LINECALLPARTYID_BLOCKED) > 0
             Then CallStateMessage "Caller ID bloccato"
  If (flags And LINECALLPARTYID_OUTOFAREA) > 0
             Then CallStateMessage "Caller out of area"
  If (flags And LINECALLPARTYID_UNKNOWN) > 0
          Then CallStateMessage "Caller ID sconosciuto"
  FindUser (Number)
 End Sub
```

AMTAPI PRO

amTapi Pro è un controllo ActiveX che può essere incorporato nelle applicazioni per effettuare e rispondere alle chiamate vocali e dati, catturare l'ID del chiamante, identificare toni DTMF remoti, geneare toni DTMF etc. amTapi Pro è un componente TAPI (Microsoft and Intel Telephony API).

La funzione *FindUser* interroga il database alla ricerca del numero telefonico catturato dall'evento *CallerId*:

```
Sub FindUser(Number As String)
  Dim strDBName As String
  Dim strConnect As String
  Set Cn = New ADODB.Connection
  Set Rs = New ADODB.Recordset
  strDBName = "TelephonBook.mdb" ' Nome del
                                          Database
  strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;
                                     Data Source="
  strConnect = strConnect & App.Path &
                                  "\" & strDBName
  Cn.Open strConnect
  Rs.CursorType = adUseClient
  Rs.LockType = adLockPessimistic
  Rs.Source = "SELECT * FROM TelephonBook where
                       number like '" & Number & "';"
  Rs.ActiveConnection = Cn
  Rs.Open
  If Rs.EOF Then ' Il numero di telefono non è stato
                                           trovato
    Rs.Close
    Rs.Source = "SELECT * FROM TelephonBook;"
    Dim NameUser
    Dim AddressUser
    Dim OtherInfoUser
    NameUser = InputBox("Inserisci il nome del
                                       chiamante.")
    AddressUser = InputBox("Inserisci l'indirizzo del
                                       chiamante.")
    OtherInfoUser = InputBox("Inserisci altre
                         informazioni sul chiamante.")
    Rs.Open
    Rs.AddNew ' Inserisce i dati relativi alla persona
                                    che ha telefonato
    Rs.Fields("Number") = Number
    If NameUser = "" Then Rs.Fields("Name") =
           "—-" Else Rs.Fields("Name") = NameUser
    If AddressUser = "" Then Rs.Fields("address") =
"—-" Else Rs.Fields("address") = AddressUser
    If OtherInfoUser = "" Then Rs.Fields("OtherInfo")
= "—-" Else Rs.Fields("OtherInfo") = OtherInfoUser
    Rs.Update
    Rs.Close
    Rs.Source = "SELECT * FROM TelephonBook
               where number like '" & Number & "';"
    Rs.Open
  End If
  'Visualizza le informazioni nella TextBox
  CallStateMessage "———————————————-"
  CallStateMessage "Numero telefonico: " &
                                Rs.Fields("Number")
  CallStateMessage "Nome: " & Rs.Fields("name")
  CallStateMessage "Indirizzo: " & Rs.Fields("address")
  CallStateMessage "Altre informazioni:
                         " & Rs.Fields("OtherInfo")
  CallStateMessage "———————————————-"
  Rs.Close
  Cn.Close
  Set Rs = Nothing
  Set Cn = Nothing
End Sub
```

Il risultato sarà la visualizzazione nella textBox *TextCallState* del nome di chi ci sta chiamando; nel caso in cui il nominativo non è presente nel database, automaticamente si apriranno delle finestre di input che permetteranno l'inserimento dei dati mancanti.

CONCLUSIONI

L'applicativo realizzato, efficace strumento di controllo e sicurezza, è inoltre finalizzato all'ottimizzazione dei tempi nell'erogazione di servizi, fruibili attraverso la linea telefonica: si pensi ad esempio alla quantità di informazioni contenute in un database di un Hotel e alla possibilità che lo strumento da noi realizzato, per l'identificazione del numero telefonico delle chiamate in entrata, diventi in un simile contesto un input per altri applicativi che gestiscono i dati.

Luigi Salerno

Un pattern per il parsing SAX di file XML

Java Legge (e interpreta) XML

L'XML è a tutt'oggi uno standard collaudato e consolidato, tanto da essere utilizzato ampiamente in quasi tutte le applicazioni.

Nel corso dell'articolo descriveremo un pattern utile nel parsing di un file XML e nella creazione degli oggetti Java che contengano le informazioni lette. Il pattern è facilmente utilizzabile in tutti i contesti in cui è necessario leggere un file XML. In questo caso mostreremo un esempio di utilizzo del pattern per un'applicazione che legge la propria configurazione da un file XML. Tale esempio potrà essere velocemente integrato in qualsiasi applicazione.

INTRODUZIONE

Da quando l'XML è diventato uno standard de facto, i metodi possibili per effettuare il parsing sono principalmente due: il DOM ed il SAX. Non entreremo troppo in dettaglio sulle caratteristiche dei due metodi. In ogni caso maggiori informazioni sono disponibili nei relativi box. Il pattern oggetto dell'articolo si basa sul SAX 2 (in particolare nel codice si è utilizzata la libreria *Xerces 2*). In tal modo si sfrutta il vantaggio, offerto dal SAX, di poter gestire anche file di grosse dimensioni. L'esempio che introdurremo si basa sulla lettura di un file XML in cui è riportata la configurazione di una generica applicazione.
L'utilizzo di un file di configurazione per una qualsiasi applicazione, piccola o grande che sia, è una problematica ricorrente, in quanto spesso è necessario impostare il valore di alcuni parametri per il corretto funzionamento dell'applicazione stessa senza intervenire sul codice. Con l'avvento di Java, i file properties si sono rivelati utilissimi soprattutto per la facilità di utilizzo. L'XML poi ha rivoluzionato la scena, in quanto ha permesso di organizzare i dati di configurazione in una struttura gerarchica semplice e flessibile, pronta a crescere in complessità quando sia necessario.

DESCRIZIONE DEL PATTERN

Il pattern che andiamo ad illustrare è basato su poche classi (vedi Fig. 1) che si occupano del lavoro di preparazione alla lettura del file XML mediante la creazione e la configurazione degli oggetti SAX necessari, permettendoci di concentrarci sulla sola gestione degli elementi XML letti. La strategia del pattern è infatti quella di leggere il file XML, gestirne gli eventi e comunicare con l'esterno mediante il metodo *readElement* dell'interfaccia *XMLParserListener*. Come illustrato in figura il pattern consta soltanto di altre due classi principali oltre all'interfaccia vista: *XMLParser* e *XMLFileReader.* Passiamo ora ad esaminare in maggior dettaglio le classi del pattern. Quando leggiamo un file XML mediante SAX, dobbiamo implementare alcune interfacce, i cui metodi verranno invocati al verificarsi di determinati eventi. L'interfaccia più importante è *ContentHandler*, la quale esporta una serie di metodi riportati nella tabella seguente:

```
public void startDocument( ) throws SAXException
public void endDocument( ) throws SAXException
public void startElement( String namespaceURI, String
        localName, String qName, Attributes attr ) throws
                                              SAXException
public void endElement( String namespaceURI, String
        localName, String qName ) throws SAXException
public void characters( char[] ch, int start, int length )
                                       throws SAXException
```

Questi metodi vengono richiamati dal parser SAX, mentre legge il file, al verificarsi di eventi quali l'apertura o chiusura del tag *document* o dei vari tag *element*. Un parser abbastanza semplice dovrebbe fornire almeno una classe che implementi tale interfaccia, contenendo nel metodo *startElement* il codice che gestisce la lettura degli attributi dell'element,

Fig. 1: ***Class diagram relativo al pattern.***

il cui tag è aperto in quel momento (un banale esempio potrebbe stampare semplicemente il nome dell'*element* fornendo così un dump della struttura del file XML). Inoltre, per utilizzare il parser SAX, bisogna creare, mediante factory, un oggetto *XMLReader* che si incarica di gestire la lettura del file (per ulteriori informazioni riferirsi all'uso del SAX) e configurarlo con alcune righe di codice.

```
System.setProperty("org.xml.sax.driver",
        "org.apache.xerces.parsers.SAXParser");
// Create SAX 2 parser...
XMLReader xr = XMLReaderFactory.createXMLReader();
xr.setFeature("http://xml.org/sax/features
                      /validation", true);
// Set the ContentHandler...
xr.setContentHandler(new MyContentHandler());
// Parse the file...
xr.parse( new InputSource(new FileReader(fileName)));
```

Come si vede, per prima cosa si setta il SAX driver utilizzato (in questo caso *Xerces 2*) e quindi si crea un oggetto *XMLReader* impostando su di esso una istanza di *ContentHandler*. Dopodiché, nell'ultima riga, inzia il parsing. Notare che l'operazione fondamentale è proprio l'impostazione del *ContentHandler* con l'implementazione (classe *MyContentHandler* in questo caso) adatta al nostro scopo, mentre il resto del codice resta più o meno invariato nella maggior parte dei casi. L'implementazione dell'interfaccia *ContentHandler* richiede di implementare tutti i metodi visti nella tabella precedente anche se fondamentalmente, nella lettura di un file XML mediante SAX, l'evento principale a cui siamo interessati è unicamente quello relativo all'apertura del tag di un element, evento gestito mediante il metodo *startElement*. Iniziamo pertanto definendo l'interfaccia *XMLParserListener* che definisce un metodo *readElement* del tipo

```
public boolean readElement(String url,Attributes attr)
                        throws SAXException;
```

Diamo innanzitutto un'occhiata ai parametri facendo prima una piccola premessa. Quando riceviamo un evento relativo all'apertura del tag di un element, il parser SAX chiama il metodo *startElement* che abbiamo implementato passando tra i vari parametri il nome dell'*element*. Spesso accade che la lettura di un file XML richieda l'interpretazione o l'elaborazione della struttura, compiendo una operazione diversa (ad esempio la creazione di un oggetto) a seconda dell'elemento corrente che ha generato l'evento. Per decidere tale operazione abbiamo a disposizione solo il nome dell'elemento, ma in realtà questa informazione non basta ad identificare in modo univoco un element all'interno di una struttura XML, in quanto diverse istanze dello stesso element possono comparire in differenti punti della struttura. A tale scopo osserviamo la struttura dell'XML riportata di seguito, dove sono descritti i dati di un ordine di un cliente, comprensivo dell'indirizzo di spedizione (elemento *DeliveryAddress*), per una serie di libri.

```
<CustomerOrder number="1285901"
                    date="15-09-2003">
<Customer cname="Robert" csurname="Palmer" />
<DeliveryAddress daddress="6539 Dumbarton Circle,
                Fremont" zipcode="94555"/>
<Position number="1">
<Book book-title="Design Patterns" />
</Position>
<Position number="2">
<Book book-title="C++ Handbook" />
</Position>
</CustomerOrder>
```

In questo caso, il nome degli element è sufficiente ad indentificare in modo univoco la posizione dell'element nella struttura. Viceversa, osserviamo la porzione di XML seguente, dove ogni posizione dell'ordine può avere associato un proprio indirizzo di spedizione differente da quello generale relativo all'ordine.

```
<CustomerOrder number="1285901" date="15-09-2003">
<Customer cname="Robert" csurname="Palmer" />
<DeliveryAddress daddress="6539 Dumbarton Circle,
                Fremont" zipcode="94555"/>
<Position number="1">
<Book book-title="Design Patterns" />
<DeliveryAddress daddress="243 Major Street,
                Fremont" zipcode="94555"/>
</Position>
<Position number="2">
<Book book-title="C++ Handbook" />
</Position>
</CustomerOrder>
```

In tale caso è utile conoscere, oltre al nome dell'element, la posizione in cui esso si trova all'interno della struttura XML. Ciò può essere realizzato conoscendo il percorso seguito rispettando la sequenza degli *element* che si visitano a partire dal root fino ad arrivare a quello corrente. Tale percorso può essere registrato in una stringa in cui sono presenti, nel giusto ordine di visita, i nomi degli element visitati separati dal punto. Ad esempio, possiamo avere un path del tipo

```
CustomerOrder.Position.DeliveryAddress
```

Tale path rappresenta il primo parametro del metodo *readElement* dell'interfaccia *XMLParserListener*, mentre il secondo è invece l'oggetto *Attributes* con-

☑ SINGLETON

Per Singleton si intende una classe tale che non permetta la creazione di più di una istanza all'interno della medesima Virtual Machine Java. Tale tipologia di classe è spesso presente nei modelli logici delle applicazioni. Il suo design è ormai ben definito nella letteratura dell'OOP tanto che si parla spesso di pattern Singleton. In Java, tale classe si realizza creando un metodo pubblico e statico che restituisce l'istanza della classe, mantenendo protetto il costruttore.

```
public class Singleton {
protected Singleton () {}
 public static Singleton
          getInstance(){
  if (instance == null) {
    instance = new
          Singleton ();
   }
   return instance;
 }
private static Config
   Object instance = null;
}
```

tenente gli attributi dell'elemento. A questo punto, passiamo ad esaminare il tipo di ritorno del metodo che è un *boolean*. Tale valore, se ritornato come *true*, comunica che l'entità *XMLParserListener* ha elaborato l'elemento. Vedremo come utilizzare questa caratteristica più avanti, quando esamineremo l'applicazione di esempio. Tutte le classi che utilizzano il pattern e che sono interessate all'apertura del tag di un element implementeranno l'interfaccia *XMLParserListener* e, in base al parametro path ricevuto, potranno determinare se sono interessate o meno alla lettura degli attributi di quell'element oppure scartare la chiamata. La prima classe del pattern, *XMLParser*, deriva direttamente da *DefaultHandler*, una classe che implementa con metodi vuoti tutte le interfacce del SAX (*ContentHandler, ErrorHandler*, ecc…). Essa può registrare, mediante il metodo *addXMLParserListener*, tutte le entità *XMLParserListener* a cui comunicherà il verificarsi dell'apertura del tag relativo ad un element.

```
public void startElement( String namespaceURI,
                          String localName, String qName,
                    Attributes attr ) throws SAXException {
  index = buffer.length();
  if(index != 0)
    buffer.append(".");
    buffer.append(localName);
    for(int k=0;k<listeners.size();k++) {
      XMLParserListener xml = (XMLParserListener)
                                        listeners.get(k);
      if(xml.readElement(buffer.toString(),attr) == true)
      return;} }
  public void endElement( String namespaceURI,
    String localName, String qName ) throws
                                        SAXException {
      buffer.delete(index,buffer.length());
      index = buffer.toString().lastIndexOf(".");
      if(index == -1)
        index = 0; }
```

Come si può notare dal codice, tale classe costruisce anche la stringa *path* dell'elemento corrente da passare come parametro a tutte le istanze *XMLParserListener* registrate. Se una chiamata al metodo *readElement* di un *XMLParserListener* ritorna *true*, significa che l'elemento è stato elaborato (o letto) e quindi non si ritiene necessario continuare nella notifica dell'evento. La classe *XMLFileReader* è responsabile invece della lettura del file mediante SAX ed è il punto centrale della struttura. Il costruttore riceve una stringa che identifica il nome (o il path) del file da leggere e crea un oggetto di tipo *XMLParser* che riceve le notifiche degli eventi relativi al *ContentHandler*. Inoltre, è presente un metodo *addXMLParserListener* che registra nell'oggetto *XMLParser* tutte le entità *XMLParserListener* interessate. Infine, è presente il metodo *readFile* che effettua la lettura ed inizia il parsing.

```
public void readFile(ErrorHandler eHandler) throws
    SAXException,IOException,FileNotFoundException {
  System.setProperty("org.xml.sax.driver",
                 "org.apache.xerces.parsers.SAXParser");
  // Create SAX 2 parser...
  XMLReader xr = XMLReaderFactory.createXMLReader();
  xr.setFeature("http://xml.org/sax/features
                                  /validation", true);
  // Set the ContentHandler...
  xr.setContentHandler(parser);
  xr.setErrorHandler(eHandler);
  // Parse the file...
  xr.parse( new InputSource(new FileReader(fileName)));
}
```

Si può notare che l'*ErrorHandler* utilizzato è atteso come parametro del metodo *readFile*. In questo modo si consente di gestire errori di parsing al di fuori del pattern. Come *ContentHandler* invece viene passato il riferimento all'oggetto *XMLParser* creato. L'uso del pattern è mostrato dal sequence diagram riportato in Fig. 2.

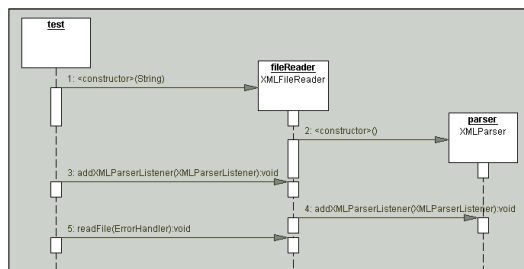
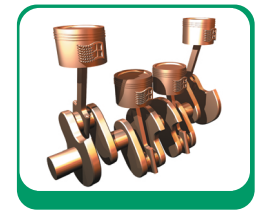



Fig. 2: ***Sequence diagram per l'uso del pattern.***

LETTURA DEL FILE CONFIGURAZIONE DI UN'APPLICAZIONE

Per il nostro esempio, immaginiamo un piccolo server RMI che esporti metodi per accedere ad un database. Possiamo definire la sua configurazione in una struttura molto semplice, mostrata di seguito.

```
<!—Specify the configuration.—>
<server-config xmlns:xsi="http://www.w3.org/2001
                                /XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="cfg.xsd">
  <!—Specify the db access configuration.—>
  <db-access db-driver= "oracle.jdbc.driver.OracleDriver"
       db-url="jdbc:oracle:thin:@kyoto:1521:ORADB"
  user="admin" password="jack" />
  <!—Specify logging.—>
  <logging debug="true" file="./out.txt"/>
  <!—It specifies the path of the policy file.—>
  <java-security policy-file="./java.policy"/>
  <!—It specifies the used port and if the RMI registry
```

☑ PARSING: DOM VS SAX

La principale differenza tra i due metodi sta nel fatto che mentre con il DOM l'intera struttura è caricata in memoria e mantenuta in una gerarchia di oggetti le cui classi sono ben definite, con il SAX al momento della lettura da file vengono generati degli eventi nell'apertura o chiusura di particolari tag o di altre condizioni.

```
 should be created within the agent process or not.—>
   <rmi-registry external="false" port="1099"/>
</server-config>
```

Sono presenti pochi elementi:

db-access, relativo alle proprietà di accesso al database (driver da utilizzare, URL del database ed utenza e password);
logging, per le informazioni sul logging in modalità di debug su file;
java-security per impostare il file di policy da utilizzare;
rmi-registry per i settaggi del registro RMI (porta e se utilizzare o meno un registro esterno al processo Java del server).

I parametri di configurazione non sono altro che gli attributi degli elementi letti, i cui valori vengono registrati in una classe che esporrà dei metodi *get* per consentire l'accesso agli elementi a chiunque li richieda. In generale però, soprattutto nel caso di file di configurazione complessi, è consigliabile partizionare la struttura XML in diverse classi di gestione ognuna relativa alla configurazione di una differente funzionalità dell'applicazione. Nel nostro caso partizioniamo il file XML in tre classi: *DbAccess, Logging* e *RMIServer.* Esse contengono degli attributi in cui registrano i parametri letti dal file XML e dei metodi get per accedervi. Inoltre tali classi implementano l'interfaccia, definita nel pattern, *XMLParserListener* in quanto devono registrare i valori dei parametri degli elementi a cui sono interessati. Vediamo in dettaglio come è fatta la classe *DbAccess* (le classi *Logging* e *RMIServer* sono concettualmente identica).

```
private final static String DB_ACCESS_ELEMENT
                                    = "db-access";
private final static String DB_DRIVER = "db-driver";
private final static String DB_URL = "db-url";
private final static String USER = "user";
private final static String PASSWORD = "password";
 public boolean readElement(String path,Attributes attr) {
if(path.endsWith(DB_ACCESS_ELEMENT)) {
 for ( int i = 0; i < attr.getLength();i++ ) {
    if(attr.getLocalName(i).equals(DB_DRIVER))
       dbDriver = attr.getValue(i);
    if(attr.getLocalName(i).equals(DB_URL))
       dbUrl = attr.getValue(i);
    if(attr.getLocalName(i).equals(USER))
       user = attr.getValue(i);
    if(attr.getLocalName(i).equals(PASSWORD))
       password = attr.getValue(i); }
  return true; }
return false;}
```

Sono innanzitutto definite delle costanti relative al nome dell'elemento e dei suoi attributi. Quindi viene implementato il metodo *readElement*. Quest'ultimo controlla che il path dell'elemento sia quello a cui siamo interessati ed in caso positivo cicla sugli attributi e li memorizza ritornando *true*. Generalmente un'applicazione legge la sua configurazione e ne valida il contenuto al momento dell'avvio, ma potrebbe utilizzare i parametri letti in punti del codice e/o momenti di esecuzione diversi. A tal fine si può utilizzare una classe *singleton* (per maggiori informazioni vedere il relativo box), *ConfigObject*, che funziona come contenitore di tutti le classi (in questo caso *DbAccess, Logging* e *RMIServer*) le quali gestiscono al loro interno i parametri della configurazione. A questo punto non resta altro che creare una semplice classe che avvii il processo di lettura del file XML ed imposti i relativi listener.

```
public class TestXML implements ErrorHandler {
```

La classe implementa l'interfaccia *ErrorHandler*, i cui metodi vengono richiamati nel caso si verifichino errori di lettura o di validazione. Nel nostro caso essi stamperanno a video l'errore interrompendo l'applicazione. Nel costruttore della classe si trova la parte più importante del codice.

```
public TestXML(String fileName) throws
SAXException,IOException,FileNotFoundException {
  System.out.println("reading file " + fileName);
  XMLFileReader reader = new
  XMLFileReader(fileName);
  reader.addXMLParserListener(
               ConfigObject.getInstance().getDbAccess());
  reader.addXMLParserListener(
                 ConfigObject.getInstance().getLogging());
  reader.addXMLParserListener(
              ConfigObject.getInstance().getRMIServer());
  reader.readFile(this);
  // dumping
  DbAccess db = ConfigObject.getInstance().getDbAccess();
  System.out.println("db-driver = " + db.getDbDriver());
  System.out.println("db-url = " + db.getDbUrl());
  Logging log = ConfigObject.getInstance().getLogging();
  System.out.println("debug = " + log.isDebug());
  System.out.println("file = " + log.getFile());
  … }
```

L'utilizzo del pattern è condensato in poche righe (come visto nel sequence diagram precedente). Viene creato un oggetto *XMLFileReader* con il nome del file XML e vengono registrati, come *XMLParserListener*, gli oggetti *DbAccess* e *Logging*. Infine viene avviata la lettura del file, impostando come *ErrorHandler* l'istanza this della classe *TestXML*. Successivamente si stampano a video tutti i parametri di configurazione letti.

David Visicchio

☑ VANTAGGI DEL PATTERN

Come abbiamo visto nell'esempio, per l'utilizzo del pattern sono richieste poche righe di codice e l'integrazione in qualsiasi applicazione si rivela veloce ed efficace. L'esempio si è basato sulla lettura di un file di configurazione espresso in XML, ma nulla vieta di applicare tale pattern anche in altri contesti. Infatti si è riscontrato che l'uso del pattern comporta grossi benefici in termini di velocità di sviluppo del codice e soprattutto della sua manutenzione.

☑ RISORSE

Java Technology & XML
http://java.sun.com/xml/

JAXP
http://java.sun.com/xml/jaxp

SAX 2.0
http://www.saxproject.org/

Xerces 2
http://xml.apache.org/xerces2-j/

Sincronizzazione dati tra SQL Server e PocketPC

Remote Data Access con PocketPC

L'utilizzo di RDA permette di effettuare operazioni remote sul Server di Database direttamente dal pocketPC e fornisce molte altre funzionalità.

Prima di descrivere in dettaglio il componete Remote Data Access (RDA), pensiamo sia essenziale metterlo a confronto con il componente di Merge Replication. Infatti, potrebbe sembrare che i due componenti di SQL Server CE 2.0 facciano le stesse cose dato che entrambi assicurano una connettività remota con SQL Server. Prima di tutto, l'esistenza di RDA è giustificata dal suo utilizzo in scenari in cui la Merge Replication non sarebbe adatta per motivi che spiegheremo nel corso della trattazione.
RDA consente, inoltre, la connettività remota alle le versioni precedenti di SQL Server. E' appena il caso di evidenziare che mentre la Merge Replication è supportata solo in SQL Server 2000, RDA è supportato dalla versione 6.5 di SQL Server.

MERGE REPLICATION E I PROBLEMI DELLA PUBBLICAZIONE

Le potenzialità offerte da RDA sono più limitate rispetto alla Merge ma offre il vantaggio di non dover costruire la pubblicazione dei dati con tutti i problemi che questa potrebbe presentare. I problemi che potrebbero aversi con una una pubblicazione dei dati in un database SQL Server potrebbero essere dovuti alla invalidazione della pubblicazione stessa per via di modifiche nelle sue impostazioni.
Ad esempio, alcune tabelle del database potrebbero non appartenere più alla pubblicazione oppure lo schema delle tabelle pubblicate potrebbe essere stato modificato per la contemporanea modifica delle specifiche di progetto. In pratica, tutte le modifiche che riguardano le tabelle della pubblicazione possono invalidarla e quindi tutti i client devono effettuare una nuova sottoscrizione dei dati. È appena il caso di evidenziare che la nuova operazione di sottoscrizione va resa del tutto trasparente all'utente e quindi gestita via codice. La gestione di una pubblicazione non valida deve riguardare il fatto che i client non si accorgano dell'errore di "Pubblicazione non valida". Infatti, effettuare una nuova sottoscrizione di una pubblicazione potrebbe comportare la perdita dei dati sul database del palmare se questi non fossero stati sincronizzati sul server prima della invalidazione. In sintesi, possiamo dire che uno dei possibili scenari in cui utilizzare la tecnologia della Remote Data Access è quando sono talmente poche le tabelle del database replicate sul palmare da non giustificare la gestione di una pubblicazione.

TIPICHE OPERAZIONI RDA

L'Object Model della tecnologia RDA utilizza l'oggetto *RemoteDataAccess* per implementare la sua logica di connettività remota. Questo oggetto dispone di tre metodi fondamentali con i quali è possibile effettuare operazioni direttamente su SQL Server oppure compiere delle modifiche dei dati sul palmare e aggiornarli in un secondo momento. Il primo metodo che prendiamo in considerazione è il metodo *SubmitSQL*. Con questo metodo possiamo inviare delle istruzioni SQL direttamente su SQL Server e quindi effettuare operazioni di inserimento, aggiornamento o cancellazione remota di records. La logica di *SubmitSQL* mette in risalto il fatto che è il PocketPC che controlla la comunicazione. Gli altri due metodi che passiamo in rassegna sono i metodi *Push* e *Pull*.
Il metodo *Pull* permette di trasferire i dati da SQL Server al database del PocketPC. Il metodo *Push* permette di trasferire i dati dal PocketPC verso il Server di Database. E' appena il caso di evidenziare che i dati sono inviati al server e inseriti nella tabella interessata dal "pushing" in modo incondizionato, modificando i dati inseriti con una precedente operazione di Pull.

☑ REQUISITI
Ingredienti Hardware: Pocket PC o (dispositivo Windows CE Based), Computer con almeno processore Pentium II e 128 MB di memoria.
Ingredienti Software: Windows 98 SE /2000/XP, IIS (oppure PWS-Personal WEB Server con Windows 9x), SQL Server 2000 con Service Pack 1 o superiore, SQL Server CE 2.0, Embedded Visual Tools.

PULL DEI DATI

Diamo un'occhiata alla procedura *PullData* che fa uso del metodo *Pull* dell'oggetto *RemoteDataAccess:*

```
Dim paco As ADOCE.Connection
Set paco = CreateObject("ADOCE.Connection.3.1")
Dim pRDA As SSCE.RemoteDataAccess
Dim strSQL As String
CONNECTION_STRING = "Provider=Microsoft.SQLSERVER.
        OLEDB.CE.2.0; Data Source=\ MioDBPalmare.sdf"
` creo il  database
 If CreateDB Then
       MsgBox ("Database creato")
On Error Resume Next
   PushData
   Set pRDA = CreateObject("SSCE.RemoteDataAccess.2.0")
   pRDA.InternetURL = "http://MioServer/
                  CartellaAgenteSynch/sscesa20.dll"
   pRDA.LocalConnectionString =
                  "Data Source=\MioDBPalmare.sdf"
   'Cancello le eventuali tabelle
   paco.Open CONNECTION_STRING
   paco.Execute "DROP TABLE " & pstrDoubleQuote &
                  "MiaTabella" & q pstrDoubleQuote
   paco.Close
   ` Effettuo il Pull della tabella
   pRDA.Pull "Mia_Tabella", "SELECT Campo1, Campo2,..
       campoN FROM MiaTabella", "Provider=sqloledb;Data
                  Source=NomeIstanzaSQLServer; Initial
                  Catalog=MioDBServer;user id=guest;
   password=", TRACKINGOFF
    If pRDA.ErrorRecords.Count > 0 Then
       ShowErrors pRDA.ErrorRecords
    End If
    Set pRDA = Nothing ` Distruggo l'oggetto RDA
End Sub
```

Nella procedura di *Pull* precedente possiamo notare il componente di *RDA pRDA* che viene istanziato con il *PROGID SSCE.RemoteDataAccess.2.0* corrispondente alla versione 2.0 di SQL Server CE. Una volta istanziato l'oggetto di RDA dobbiamo impostare le sue property:

1) InternetURL: rappresenta il percorso dell'agente di sincronizzazione *sscesa20.dll.* Ovviamente, poichè la connettività remota avviene tramite protocollo HTTP allora dobbiamo fare in modo che il computer *MioServer,* che agisce da server di *Database,* abbia installato IIS come WEB Server e sia stata opportunamente configurata la cartella *CartellaAgenteSynch* per rendere accessibile dai client l'agente di soncronizzazione;

2) LocalConnectionString: la stringa di connessione relative alla istanza del database sul palmare.

Nella procedura possiamo notare l'invocazione del metodo *CreateDB;* lo scopo del metodo è quello di creare sul palmare il file di database nella posizione specificata se questo non esistesse. Ecco una possibile implentazione di *CreateDB:*

```
Function CreateDB() As Boolean
  On Error Resume Next
   Dim paca As ADOXCE.Catalog
   Set paca = CreateObject("ADOXCE.Catalog.3.1")
   paca.Create CONNECTION_STRING
   CreateDB = True
   Set paca = Nothing
End Function
```

Impostate le proprietà del componente si può invocare la *Pull* passandovi il comando SQL di *SELECT* per selezionare i dati della tabella *MiaTabella*; il *Data Source,* ovvero il nome del computer su cui è installato SQL Server; *Initial Catalog* che rappresenta il nome del *Database*; infine la *password* e la *userid* dell'utente anonimo che si connette al computer remoto con la tecnologia RDA. Precisiamo che l'operazione di *Pull* permetterà di creare nel File di database *MioDBPalmare.sdf* lo schema della tabella *MiaTabella* presente sul server insieme ai dati compatibili con la query di selezione passata come primo parametro del metodo *PULL.* L'operazione di *PULL* non andrà a buon fine nel caso in cui il File di Database già contenesse la tabella *MiaTabella,* per cui è necessario un'operazione di *DROP* utilizzando l'oggetto connessione paco di *ADOCE 3.1.* Fatta questa considerazione, comprendiamo perché prima della *PULL* venga invocata la procedura *PushData. PushData* utilizzerà il metodo *Push* dell'oggetto *RemoteDataAccess* per inviare i dati al server. Ecco una possibile implementazione della procedura:

```
Dim pRDA As SSCE.RemoteDataAccess
CONNECTION_STRING = "Provider=Microsoft.
                  SQLSERVER.OLEDB.CE.2.0; Data
                        Source=\MioDBPalmare.sdf"
On Error Resume Next
   Set pRDA = CreateObject("SSCE.RemoteDataAccess.2.0")
   pRDA.InternetURL = "http://MioServer/
                  CartellaAgenteSynch/sscesa20.dll"
   pRDA.LocalConnectionString = "Data Source=
                              \MioDBPalmare.sdf"
    pRDA.Push "MiaTabella", " Data Source=
              NomeIstanzaSQLServer; Initial Catalog=
                 MioDBServer;user id=sa;password="
   If pRDA.ErrorRecords.Count > 0 Then
       ShowErrors pRDA.ErrorRecords
  End If
  Set pRDA = Nothing
End Sub
```

In un prossimo articolo vedremo come applicare la logica ad un caso reale.

Elmiro Tavolaro

☑ REMOTE DATA ACCESS

Rappresenta una tecnologia alternativa alla Merge Replication in grado di supportare connettività remota a client PocketPC con SQL Server CE a computer con installato SQL Server. Rispetto alla Merge Replication ha due sostanziali vantaggi:1) Assenza della gestione di una pubblicazione sul server; 2) maggiore velocità poichè vengono inviati una quantità di dati inferiore.

☑ RDA, QUANDO?

RDA dovrebbe essere usata in scenari in cui i dati possano essere partizionati in maniera univoca tra diversi utenti in modo tale che non accadano mai i conflitti.
I conflitti possono accadere nel caso che i client pocketPC modifichino gli stessi record di una o più tabelle.

Il palmare accede al DB aziendale

Negli articoli precedenti abbiamo lavorato molto sulla multicanalità applicativa, cercando di capirne i concetti di base e sviluppando insieme una completa applicazione multicanale. Estendiamola verso dispositivi J2ME.

L'applicazione multicanale, realizzata con tecnologia J2EE ed utilizzando il Pattern MVC, consente allo stato attuale il dispatching dei servizi su canali tradizionali come HTML su HTTP e WML su Wap, ma comprende anche una completa integrazione verso il mondo dei Web Services su SOAP. Attualmente quindi è possibile consumare i servizi applicativi sia da un tradizionale client sia in maniera trasparente da applicazioni terze che, con l'obiettivo di erogare i propri servizi, utilizzano in maniera trasparente quelli messi a disposizione dalla nostra applicazione. A questo punto non ci resta che identificare l'ultimo dei canali che intendiamo far gestire alla nostra applicazione ed aggiungerlo ai meccanismi di erogazione, si tratta del canale J2ME.

OBIETTIVO

Quello che vogliamo ottenere è che la nostra applicazione multicanale possa erogare i suoi servizi di ricerca all'interno della rubrica telefonica anche ad una qualsiasi applicazione Java 2 Micro Edition che ne faccia richiesta attraverso una chiamata HTTP.
In questo modo saremo in grado di scrivere un'applicazione J2ME che possa girare indifferentemente su palmari o telefoni cellulari compatibili e che, dopo essersi collegata ad una rete attraverso un protocollo di comunicazione come GPRS, possa effettuare ricerche all'interno della nostra rubrica telefonica. Per raggiungere questi risultati dovremo operare su due fronti distinti: innanzi tutto aggiungere il canale J2ME all'applicazione e poi sviluppare una Midlet (applicazione J2ME) che sia in grado di utilizzare il nuovo canale.

IL CANALE J2ME

L'architettura software della nostra applicazione multicanale, basata sul Pattern MVC, prevede che esista una componente di dispatching delle chiamate che in funzione della tipologia del canale utilizzato dal dispositivo chiamante effettui una ridistribuzione della chiamata stessa alla componente software che implementa la logica di business per quello specifico canale. Questa componente è una Servlet implementata dalla classe *Controller.java* la cui struttura è descritta in Fig. 1.

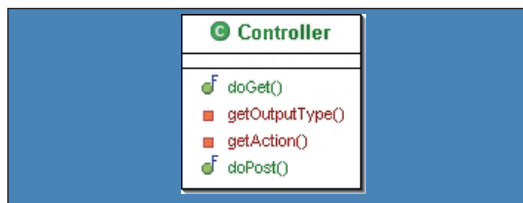



Fig. 1: **La struttura della classe Controller.**

Come si può vedere il controller è dotato del metodo privato *getOutputType()* che consente di recuperare le informazioni sul canale in base al valore dell'header HTTP "User-Agent". Modifichiamo l'implementazione di questo metodo in modo che sia in grado di riconoscere ed identificare la stringa "MIDP" e di conseguenza gestire diversamente questo canale

```
private String getOutputType(HttpServletRequest request)
{  String userAgent = request.getHeader("User-Agent");
   System.out.println("userAgent: " + userAgent + "\n");
   if (userAgent.indexOf("UP.Browser") >= 0) {
      return "wml"; }
   if (userAgent.indexOf("MIDP") >= 0) { return "midp"; }
   return "html";}
```

A questo punto il Controller è in grado di gestire correttamente un nuovo canale identificandolo come quello utilizzato dai dispositivi che possiedono la stringa "MIDP" all'interno dell'header HTTP "User-Agent".

IL SERVIZIO DA ESPORRE

Per come funziona la nostra applicazione sappiamo che la tipologia del servizio che il consumatore richiede è identificata dal parametro *Action* che viene inviato al controller e sappiamo anche che la produzione dei dati da inviare come risposta è a carico di una pagina JSP la cui nomenclatura è nella forma *action_canale.jsp*. Poiché in un'applicazione J2ME possiamo demandare parte dell'interfaccia all'applicazione stessa più di quanto non si possa fare, per esempio, con un'applicazione Web, possiamo decidere che l'unico servizio applicativo che ci interessa utilizzare da remoto sia una lista dettagliata di tutti i nominativi conformi ad un certo criterio di ricerca, quindi il servizio che andremo ad implementare sarà quello identificato dal parametro *Action=lista*. Dovremo quindi implementare la pagina *lista_midp.jsp* che avrà il seguente contenuto:

```
<%@ page contentType="text/html; charset=
  iso-8859-1" language="java" import="java.util.*" %>
<%@ taglib uri="http://www.mcapp.com/taglib"
                          prefix="mcapptags" %>
<% String cognome = (String)request.getParameter
                          ("cognome");%>
<mcapptags:ListTag language="midp"
                 cognome="<%=cognome%>"/>
```

Come si può verificare guardando il codice di questa pagina JSP l'intera pagina demanda la produzione del proprio contenuto al custom tag *ListTag* che ben conosciamo e che viene utilizzato dall'applicazione per generare i contenuti per tutti i canali. I parametri che vengono passati al tag sono il canale, che sappiamo essere *"midp"*, ed il cognome, cioè il criterio di ricerca per effettuare la query sul DB. Per il resto la pagina non si occupa di impaginare in alcun modo i dati, ma ne effettua una semplice distribuzione al dispositivo chiamante.

☑ IL PATTERN MVC

Uno degli obiettivi del paradigma Object Oriented è la possibilità di realizzare classi che possano vivere autonomamente in qualsiasi contesto e non solo in quello nel quale vengono realizzate. Questo implica che le componenti di un'applicazione debbano essere sufficientemente separate tra loro e che possano essere sostituite con altre implementate diversamente a condizione che ne rispettino l'interfaccia. Per realizzare applicazioni che soddisfino questi requisiti si utilizza il pattern MVC che consente di separare tra loro le componenti applicative: il Model che implementa le funzionalità di business, la componente di View che implementa la logica di presentazione ed il Controller che implementa la logica di controllo.

L'IMPLEMENTAZIONE

Come sappiamo il custom tag *ListTag* è implementato attraverso la classe *ListTag.java* che si occupa di fare la query sul database utilizzando come criterio di ricerca una stringa da ricercare nel campo cognome sul database della rubrica. Per fare in modo che questa classe implementi correttamente anche le richieste dal canale "midp" dovremo aggiungere una porzione di codice:

```
if (getLanguage().equals("midp")) {
   output += rs.getString("Cognome") + " "
                  + rs.getString("Nome") + "\n";
   output += "email: " + rs.getString("Email") + "\n";
   output += "tel: " + rs.getString("TelefonoFisso") + "\n";
   output += "cell: " + rs.getString(
                     "TelefonoCellulare") + "\n";
   output += "filiale: " + rs.getString("Filiale") + "\n";
   output += "unità: " + rs.getString(
                   "UnitaOrganizzativa") + "\n";
   output += "\n"; }
```

A questo punto abbiamo concluso l'implementazione del nuovo canale, rivediamo la sequenza di passi che abbiamo compiuto:

- **Modifica del Controller perché riconosca i dispositivi che utilizzano il canale J2ME**
- **Scrittura della pagina JSP che spedisce i dati al dispositivo chiamante**
- **Modifica del custom tag per renderlo compatibile con il nuovo canale**

A questo punto abbiamo terminato le modifiche all'applicazione multicanale e possiamo concentrarci sulla realizzazione dell'applicazione J2ME.

IL WIRELESS TOOLKIT

Per realizzare la nostra Midlet utilizzeremo il J2ME Wireless Toolkit 1.0.4 che implementa le specifiche 1.0 del profilo MIDP. L'interfaccia del Wireless Toolkit consiste sostanzialmente in un tool chiamato *KToolbar* visibile in Fig. 2 che consente la creazione e la gestione di interi progetto J2ME e la produzione automatica di tutti i file di configurazione e struttura che servono per la realizzazione di una Midlet.

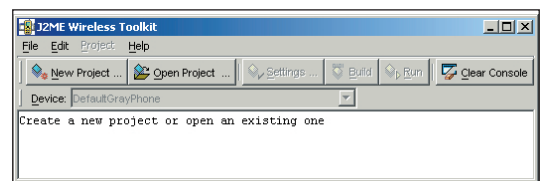



Fig. 2: La KToolbar del J2ME Wireless Toolkit.

Per creare una nuova Midlet è sufficiente selezionare il pulsante *"New Project"* ed indicare il nome del progetto (nel nostro caso *TestRubrica*) ed il nome della classe che implementa la Midlet principale dell'applicazione (nel nostro caso *RubricaTelefonica*). Appena premuto il pulsante *"Create Project"* il toolkit genererà automaticamente l'alberatura delle directory deputate a contenere tutti i file della nostra applicazione. L'alberatura ha la seguente struttura a

partire dalla directory di installazione del Wireless Toolkit:

apps\TestRubrica\bin: contiene il file di manifesto dell'applicazione ed il file JAD di descrizione della Midlet
apps\TestRubrica\classes: contiene le classi compilate che implementano la Midlet
apps\TestRubrica\lib: contiene eventuali librerie esterne necessarie al funzionamento dell'applicazione
apps\TestRubrica\res: contiene file di risorsa come le immagini utilizzate nell'applicazione
apps\TestRubrica\src: contiene i sorgenti delle classi dell'applicazione

LA MIDLET

Sappiamo che la Midlet che dobbiamo implementare si deve chiamare *RubricaTelefonica.java* e sappiamo che questo file deve essere posizionato nella directory *src* dell'alberatura dell'applicazione, ma cosa ci aspettiamo da questa applicazione? Il nostro obiettivo funzionale è realizzare un'applicazione business che possa essere scaricata ed installata sui cellulari o sui palmari aziendali dei dipendenti e che consenta di effettuare ricerche all'interno della rubrica aziendale. Per prima cosa quindi dovremo implementare il metodo *startApp()* della Midlet che viene eseguito quando il controllore dei processi della macchina virtuale inizia l'esecuzione dell'applicazione. Ecco il codice del metodo:

```
public void startApp() throws
MIDletStateChangeException {
  display = Display.getDisplay(this);
  menu = new List("Rubrica Aziendale", Choice.IMPLICIT);
  menu.append("Ricerca cognome", null);
  menu.addCommand(exitCommand);
  menu.setCommandListener(this);
  mainMenu(); }
```

La prima cosa che viene fatta all'interno del codice della Midlet quindi è costruire un oggetto di interfaccia istanza della classe *Display*. Si tratta di un oggetto appartenente al package *javax.microedition.lcdui* che si occupa di gestire lo schermo del dispositivo attraverso API ad alto livello. Questo è di fatto il menù di primo livello della nostra Midlet, un menù dotato della sola voce *"Ricerca cognome"*.
Il listener dei comandi è realizzato attraverso l'invocazione di un oggetto *Command* su un oggetto visualizzabile *(Displayable)* ed il codice che implementa questo comportamento è il seguente:

```
public void commandAction(Command c, Displayable d) {
    String label = c.getLabel();
    if (label.equals("Exit")) {destroyApp(true);}
    else if (label.equals("Back")) {mainMenu();}
    else if (label.equals("Submit")) {
    user = input.getString();
    try {invokeServlet(url + user);}
    catch(IOException e) {}
    } else {addName();} }
```

La pressione del tasto *Submit* del dispositivo in corrispondenza dell'unica voce di menù causa quindi l'esecuzione del metodo *addName()* che si occupa di raccogliere il criterio di ricerca che sarà poi passato all'applicazione multicanale.
Il metodo *addName()* è fatto in questomodo:

```
public void addName() {
    input = new TextBox("Inserisci il cognome:", "", 5,
                                        TextField.ANY);
    input.addCommand(submitCommand);
    input.addCommand(backCommand);
    input.setCommandListener(this);
    input.setString("");
    display.setCurrent(input); }
```

e consente di visualizzare un oggetto di input sul display del dispositivo per fare in modo che l'utente possa inserire la stringa da ricercare, quando l'utente preme il tasto di submit del dispositivo (verosimilmente dopo aver inserito il cognome da ricercare) la stringa che ha inserito viene memorizzata e viene successivamente appesa all'url da raggiungere in una chiamata HTTP. La chiamata viene instradata all'applicazione remota attraverso il metodo *invokeServlet(String url)* che si occupa di effettuare una Request HTTP e stampare a video il pacchetto di byte ricevuto nella Response. Per effettuare la connessione remota si utilizza un oggetto *Connector* che prende in input un parametro stringa che identifica la URL da raggiungere. La URL dell'applicazione multicanale è memorizzata in una stringa esterna ed è valorizzata in questo modo:

```
String url = "http://localhost:8080/mcapp
                    /Controller?Action=lista&cognome=";
```

Come si vede l'URL è esattamente quello dell'applicazione multicanale che gira (in questo caso) in locale ed a questa viene passato il parametro *Action=lista* che identifica il servizio che ci aspettiamo dall'applicazione stessa. Al Controller viene anche passato, per poter effettuare la ricerca, il parametro *cognome=* cui viene appeso il testo che l'utente ha inserito all'interno del textbox della Midlet.
In questo modo l'applicazione multicanale potrà effettuare la query sul database e restituire, secondo le modalità che abbiamo visto in precedenza, il pacchetto di dati al dispositivo chiamante che non dovrà far altro che visualizzarlo all'interno dell'og-

☑ J2ME

**Sun Microsystem propone tre diverse architetture software basate sulla tecnologia Java, queste sono indirizzate ad ambienti e a dispositivi differenti.
Si tratta di J2SE (Java 2 Standard Edition), J2EE (Java 2 Enterprise Edition) e J2ME (Java 2 Micro Edition).
Java 2 Micro Edition è un sottoinsieme della piattaforma Java 2 Standard Edition alleggerito ed ottimizzato per essere compatibile con il maggior numero di dispositivi elettronici di consumo.
L'architettura di J2ME è composta da moduli tra loro ortogonali: le configurazioni ed i profili.
Le configurazioni determinano quale virtual machine sarà utilizzata su una particolare classe di dispositivi e poi, più nel dettaglio, i profili determinano quali moduli applicativi potranno essere utilizzati all'interno di una particolare configurazione.
L'architettura generale della piattaforma J2ME è visibile in figura 2.
Le applicazioni J2ME prendono il nome di Midlet e possono essere eseguite su qualsiasi dispositivo per il quale esista una macchina virtuale J2ME.**

getto *Display*. L'ultima cosa che ci rimane da fare è implementare il metodo *setRequestHeaders(HttpConnection c)* che si occuperà di valorizzare gli header della chiamata HTTP secondo lo schema classico delle Midlet, cosa che non viene fatta, e questo può essere un vantaggio, in automatico dalla macchina virtuale J2ME. Ecco il codice di questo metodo:

```
void setRequestHeaders(HttpConnection c) throws
                                    IOException {
    String conf = System.getProperty(
                    "microedition.configuration");
    String prof = System.getProperty("microedition.profiles");
    String locale = System.getProperty(
                    "microedition.locale");
    String ua = "Profile/" + prof + " Configuration/" + conf;
    c.setRequestProperty("User-Agent", ua);
    if (locale != null) {
        c.setRequestProperty("Content-Language", locale); }
}
```

Il valore dello *User-Agent* per la nostra Midlet sarà quindi il seguente:

```
Profile/MIDP-1.0 Configuration/CLDC-1.0
```

TESTIAMO L'APPLICAZIONE

Una volta inserito il codice della Midlet all'interno del file *RubricaTelefonica.java* possiamo compilare l'intera Midlet attraverso il pulsante *Build* presente sulla KTollbar. Il risultato dovrebbe essere *"Build Complete"* che identifica il successo della compilazione senza errori. A questo punto possiamo selezionare tra quelli disponibili il dispositivo all'interno del quale emulare la Midlet e premere il pulsante *Run*, otterremo l'esecuzione dell'emulatore con la Midlet già pronta. In Fig. 3 la nostra applicazione J2ME eseguita all'interno dell'emulatore *Default Color Phone* e ci permette di selezionare l'unica Midlet a disposizione, cioè *TestRubrica*. A questo punto la Midlet parte (Fig. 4) e viene mostrato il menù di primo livello nel quale selezioniamo l'unica voce disponibile, cioè la funzionalità di ricerca per cognome all'interno della rubrica aziendale. Quello che ci aspettiamo è che ci venga consentito di inserire il criterio di ricerca ed infatti, come se vede in Fig. 5, la Midlet presenta un campo di testo nel quale inserire il cognome da ricercare. Premendo il pulsante di *Submit* otterremo l'esecuzione della chiamata all'applicazione multicanale che sarà in grado di riconoscere il canale J2ME e ci restituirà la lista di tutti i nominativi compatibili con il criterio di ricerca che abbiamo inserito. Questa lista verrà presentata direttamente sul display del telefonino e sarà navigabile direttamente come si può vedere in Fig. 6. Naturalmente la nostra Midlet si può far girare su qualsiasi dispositivo compatibile J2ME, sia esso un telefono cellulare, un palmare, uno smartphone, un orologio, una macchina fotografica o qualsiasi dispositivo dotato della possibilità di connettersi alla rete sulla quale si trova in esecuzione l'applicazione multicanale che fornisce il servizio. In Fig. 7, per esempio, il risultato della ricerca effettuata su un Palm m515, dispositivo per il quale esiste un'ottima implementazione del profilo MIDP.

Massimo Canducci

Fig. 3: Lanciamo la nostra Midlet.
Fig. 4: Selezione della voce di menu.

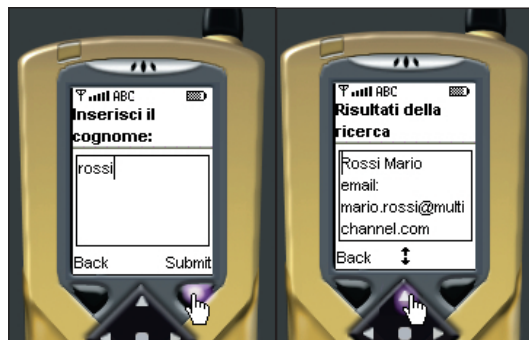



Fig. 5: Inserimento dei criteri di ricerca.
Fig. 6: I risultati della ricerca restituiti dal servizio remoto.

Fig. 7: Gli stessi risultati su un palmare collegato al servizio remoto.

☑ RIASSUMENDO

Con questa serie di articoli abbiamo analizzato a fondo lo sviluppo di applicazioni multicanale toccando aspetti di architettura come il Pattern MVC e la piattaforma J2EE e ragionando su tematiche concettuali anche complesse come il dispositivo consumatore, il protocollo di erogazione ed il canale. Siamo partiti dal realizzare un'applicazione in grado di erogare servizi sui canali Web e Wap, poi con Axis abbiamo reso i servizi applicativi dei veri e propri Web Services. In questa ultima puntata abbiamo aggiunto il canale mobile di nuova generazione, J2ME, indipendente dalla piattaforma, dal dispositivo e dalla modalità di comunicazione, sia essa GSM, GPRS, UMTS o 4G. Non resta che sperimentare e realizzare le proprie architetture tenendo sempre presente i principi di massima flessibilità senza legarsi troppo al canale di erogazione e tenendo presente che il mobile è dietro l'angolo e le nostre applicazioni devono essere pronte a supportarlo senza essere stravolte.

Testiamo la nostra abilità nel risolvere alcuni problemi

Dalla 'a' alla 'z'

E' venuto il momento di applicare quello che hai imparato nei mesi scorsi a qualche problema un po' più interessante. Ma prima dovrai sottoporti alla solita badilata di cultura e imparare un po' di roba nuova.

Riassunto della puntata precedente. Tanto per darti una rinfrescata, ti propongo un esercizio molto semplice:

Esercizio 1: *Scrivi un programma che usa un ciclo **while** per stampare un conto alla rovescia da 10 a 0.*

Se ricordi bene l'articolo del mese scorso non dovresti avere difficoltà. Ecco una soluzione:

```
class Countdown {
  public static void main(String[] args) {
    int i = 10;
    while(i >= 0) {
      System.out.println(i);
      i--;
    }
  }
}
```

La condizione del *while* è che i sia maggiore o uguale a 0 (questo è il significato dell'operatore '>='). L'*operatore di decremento*, che si scrive con un doppio segno "meno", è il compagno di quello di incremento che hai conosciuto il mese scorso. L'istruzione *"i--"* equivale quindi a *"i=i-1"*. Il resto del codice dovrebbe esserti familiare.
Diamo un'occhiata al ciclo, per essere sicuri di non aver sbagliato niente. Appena entriamo nel ciclo la variabile i vale 10. Quindi il programma stampa "10" e decrementa *i*. Cosa succede quando stiamo per uscire dal ciclo? Quando i vale 1, il valore viene stampato e *i* diventa 0. La condizione del ciclo *(i>=0)* è ancora soddisfatta. Quindi il programma entra nel ciclo un'ultima volta, stampa "0" e decrementa *i*, che ora vale -1. A questo punto la condizione del ciclo non è più soddisfatta, perché *i* è minore di 0. Quindi il programma esce dal ciclo.
Abbiamo ragionato sulle *condizioni limite* del ciclo: cosa succede quando il ciclo inizia, e cosa succede quando il ciclo sta per finire. Questo modo di ragionare ti diventerà familiare. Dovresti ragionare sulle condizioni limite ogni volta che scrivi un ciclo. Se non lo fai ti sarà facile sbagliare. Ad esempio potresti trovarti ad usare un "minore" dove invece avresti dovuto usare un "minore o uguale".
L'istruzione *while* è per ora l'unico modo che conosci per costruire un ciclo. In alcuni casi è il modo ideale. Ad esempio puoi usare un ciclo *while* per dire facilmente cose come: "resta in ascolto su questa porta di comunicazione fino a quando non ricevi un comando di questo tipo". Ma quando il ciclo si basa su un semplice contatore che viene incrementato o decrementato, come quello che hai appena visto, ti conviene di solito usare un'altra istruzione: l'istruzione *for*.

PASSO PER PASSO

La sintassi dei cicli *for* non è particolarmente bella da vedere. Purtroppo ci tocca tenercela così com'è (è una sintassi che deriva dal linguaggio C, che non è noto per la sua leggibilità). Un ciclo *for* è fatto così:

for*(inizializzazione; condizione; passo)*
istruzione

L'*inizializzazione* consiste di solito nella definizione/inizializzazione di una variabile. La *condizione* è quella che deve essere soddisfatta per entrare nel ciclo. Il *passo* è un'istruzione che viene eseguita alla fine di ogni ciclo. Ad esempio, ecco un ciclo che stampa il contenuto di un array di nome *a*:

```
for(int i = 0; i < a.length; i++)
  System.out.println(a[i]);
```

L'*inizializzazione* definisce una variabile *i* e le assegna il valore 0. Il passo incrementa *i* al termine di ciascun ciclo. La *condizione* dice: "esegui questo ciclo fintanto che *i* è minore della lunghezza dell'array *a*".
Ragioniamo sulle condizioni limite. All'inizio *i* vale 0, quindi viene stampato l'elemento *a[0]*. L'ultimo elemento che viene stampato è *a[a.length - 1]*, cioè

OBIETTIVI DI QUESTA LEZIONE

Ecco cosa ti aspetta questo mese:

Farai la conoscenza di una nuova istruzione per costruire cicli.

Studierai il meccanismo del casting.

Ti cimenterai con qualche esercizio abbastanza impegnativo.

Ti consiglio di seguire questo articolo davanti a un computer. Usa un computer soprattutto quando cerchi di risolvere gli esercizi, che ormai sono troppo complicati per risolverli "a mente".

☑ **NON SEMPRE SI PUÒ**

Non sempre è possibile fare un cast, anche se esplicito. Ad esempio Java non permette di fare nessun tipo di cast tra il tipo boolean e qualsiasi altro tipo. Un boolean è sempre un boolean, e non lo si può convertire:

```
int x = (int)true;  // errore
boolean b = (boolean)0;
                    // errore
```

☑ **PERDERE PEZZI**

Non è un caso se il compilatore non si assume la responsabilità di alcuni casting e ci costringe ad usare un cast esplicito: in questi casi, infatti, è possibile perdere informazioni. Ad esempio:

```
System.out.println((int)1
                     2.9);
```

Questa riga stampa sullo schermo il numero 12, che è il risultato della conversione forzata del double 12.9 al tipo *int*. Nota che il numero non è stato arrotondato all'intero più vicino, ma semplicemente e brutalmente troncato.

l'ultimo elemento dell'array (se la lunghezza di *a* è 10, il suo ultimo elemento è *a[9]*). Subito dopo *i* diventa uguale ad *a.length*, quindi la condizione non è più soddisfatta e si esce dal ciclo. Se l'array avesse lunghezza nulla, la condizione non sarebbe mai verificata (sia *i* che *a.length* varrebbero 0), e tutto il ciclo verrebbe semplicemente ignorato. Il ciclo è composto da una sola istruzione, ma naturalmente puoi sempre usare un blocco delimitato da parentesi graffe per eseguire più istruzioni per ciascun "passo".
Puoi sempre scegliere tra *for* e *while*, a seconda della forma che ti sembra più chiara e leggibile in ciascuna circostanza. Il ciclo che hai appena visto scritto equivale quasi esattamente a questo (tra poco ti spiegherò perché ho detto "quasi"):

```
int i = 0;
while(i < a.length) {
  System.out.println(a[i]);
    i++;
}
```

In questo caso la versione del ciclo che usa il *for* è probabilmente più leggibile di quella che usa il *while* – almeno per chi è abituato a leggere entrambe le sintassi.
L'unica differenza tecnica tra le due forme di questo ciclo ha a che fare con la visibilità delle variabili. Nel caso del ciclo *for*, la variabile i viene definita all'interno del ciclo (il contenuto delle parentesi tonde si considera una parte del ciclo). Quindi la sua visibilità si estende solo al ciclo e la variabile esce dallo "scope" alla fine del ciclo. Se provi a leggere il valore di *i* al termine del ciclo il compilatore dichiarerà di non conoscere la variabile:

```
for(int i = 0; i < a.length; i++)
  System.out.println(a[i]);
System.out.println(i);  // errore!
```

Nel caso del *while*, invece, la variabile *i* è definita all'esterno del ciclo, quindi resta visibile anche quando il ciclo è finito.

Esercizio 2: *Riscrivi il programma* ***Countdown*** *usando un ciclo* ***for*** *invece del ciclo* ***while****.*

UN TIPO DI CARATTERE

Ora vorrei usare l'istruzione *for* per risolvere un nuovo problema: stampare tutte le lettere dell'alfabeto in sequenza. Ma per fare questo devo prima parlarti un po' del tipo *char*, che finora non abbiamo mai usato. Questa digressione ci porterà a toccare un argomento nuovo e importante: il *casting*.

Se hai buona memoria, forse ricordi che abbiamo rapidamente introdotto il tipo *char* (carattere) un paio di mesi fa. Le costanti carattere si mettono tra singoli apici:

```
char c = 'x';
```

Ricorda che un carattere non è una stringa di lunghezza uno; è proprio un tipo a sé stante. Una cosa interessante è che ciascun carattere è rappresentato da un codice *Unicode*. Nello standard Unicode, a ciascun "codice" (un numero intero tra 1 e 65535) corrisponde un carattere univoco.
Dato che un carattere è anche un codice numerico, possiamo usarlo in un'espressione aritmetica:

```
System.out.println('a' + 1);
```

Questa istruzione stampa un risultato inatteso: il numero 98. Perché? Cerchiamo di capire come ragiona il sistema. L'espressione somma il carattere *'a'*, che nello standard Unicode corrisponde al codice 97, con il numero intero 1. Come si fa a sommare un carattere ad un numero? In queste situazioni il sistema deve operare una conversione di formato, cioè trasformare un valore da un tipo ad un altro tipo. Nel nostro esempio trasforma il primo operando dal tipo *char* al tipo *int* (usando il suo codice di carattere Unicode) e fa la somma.
Questa conversione automatica avviene anche in molti altri casi. Ad esempio, se chiedi a Java di concatenare una stringa con un numero:

```
System.out.println("10" + 12);
```

Qual è il risultato di questa stampa? Il sistema avrebbe due possibilità: o trasforma la stringa in un intero, o trasforma l'intero in una stringa. Il risultato sarebbe completamente diverso nei due casi, anche perché il simbolo '+' può rappresentare due operatori diversi a seconda del contesto: quando gli operandi sono stringhe, il '+' rappresenta l'operatore di concatenazione tra stringhe; quando gli operandi sono numerici, il '+' rappresenta l'operatore di somma. Quindi se il sistema decidesse di convertire la stringa in un numero il risultato stampato sullo schermo sarebbe *22*; se invece decidesse di convertire il numero in una stringa il risultato sarebbe *"1012"*. Se vuoi puoi fare tu stesso l'esperimento, ma posso dirti subito che il risultato è *"1012"*. Quindi il sistema converte il numero in una stringa. Per quale motivo Java fa sempre questa scelta e non quella opposta? Prima di tutto perché capita più spesso di convertire i numeri in stringhe (ad esempio per stampare dei risultati sullo schermo) che non viceversa. Pensa a un caso comune come questo:

```
System.out.println("La variabile x vale: " + x);
```

Ma esiste anche un altro motivo, forse più importante: mentre è sempre possibile convertire un numero in una stringa, non sempre è possibile fare il contrario (pensa ad una stringa che contiene dei caratteri alfabetici). Quindi il comportamento standard di Java è quello di convertire i numeri in stringhe nelle espressioni "miste".
Questa conversione automatica da un tipo all'altro è una caratteristica importante del linguaggio, e la vedrai diventare sempre più importante man mano che imparerai cose nuove. La conversione di tipo si chiama *casting*. Quando viene fatta automaticamente da Java, come nel caso che abbiamo visto, si chiama *cast implicito*.

TRA DUE TIPI

Torniamo ai caratteri. Vogliamo costruire un array che contiene le 26 lettere dell'alfabeto anglosassone. Possiamo riempire l'array con un ciclo *for*:

```
char[] alfabeto = new char[26];
for(int i = 0; i < alfabeto.length; i++){
  <...>
}
```

Come facciamo ad assegnare a ciascun elemento dell'array il carattere corrispondente? Possiamo approfittare del fatto che nello standard Unicode le lettere dell'alfabeto sono già nella sequenza giusta. In realtà esistono due sequenze distinte, una per le lettere minuscole (che parte da *'a'*) e una per le maiuscole (che parte da *'A'*). Noi useremo le minuscole. Una semplice soluzione può essere quella di assegnare all'elemento *alfabeto[i]* il carattere ottenuto sommando il valore di *i* al codice del carattere *'a'*. In questo modo l'elemento *alfabeto[0]* avrà valore *'a'*, alfabeto*[1]* avrà valore *'b'* e così via fino ad *alfabeto[25]*, che avrà valore *'z'*. Proviamo:

```
for(int i = 0; i < alfabeto.length; i++){
  alfabeto[i] = 'a' + i;  // errore!
}
```

Il compilatore mi ha dato un inatteso errore di compilazione. Perché?
Il problema sta nel casting. Nell'espressione *'a' + i*, il *char* viene convertito automaticamente in un *int* prima di fare la somma. Il risultato della somma dei due *int* è un *int*. Fin qui tutto bene. A questo punto, però, l'intero risultante viene assegnato ad una variabile di tipo *char* (un elemento dell'array *alfabeto*). Ora, alcuni casting vengono fatti automaticamente. Altri invece no. Ad esempio, mentre la conversione da *char* a *int* è automatica, la conversione da *int* a *char* non lo è. Questo perché è sempre possibile convertire un carattere (che al massimo vale 65535) in un intero, ma non sempre è possibile convertire un intero (che può valere decine di milioni) in un carattere. Quindi quando converto un *int* in *char* potrei superare i limiti del tipo *char*. Il compilatore non può sapere quale valore assumeranno le espressioni in fase di esecuzione, quindi per sicurezza mi ferma sempre. Questo principio vale in molte altre circostanze:

```
float f = 12.0; // errore
```

Le costanti numeriche con la virgola sono interpretate dal compilatore come dei *double*. Se cerchi di infilare un *double* in un meno capiente *float*, il compilatore ti avvisa che rischi di perdere informazioni. Lo stesso vale, ad esempio, se cerchi di mettere un valore *long* in una variabile *int*. Come risolvere questo problema? Visto che il compilatore "non si assume responsabilità", questa responsabilità ricade su noi programmatori. Dobbiamo essere noi a sapere che il valore assegnato non può superare i limiti della variabile alla quale lo assegniamo. Se sai che la conversione è innocua (o se non ti interessa perdere parte delle informazioni) puoi forzarla con un *cast esplicito*. Ecco quale sintassi devi usare:

```
float f = (float)12.0; // OK (casting esplicito)
```

Quindi possiamo riempire l'array in questo modo:

```
char[] alfabeto = new char[26];
for(int i = 0; i < alfabeto.length; i++)
  alfabeto[i] = (char)('a' + i);
```

Nota che ho dovuto mettere tra parentesi anche tutta l'espressione *('a' + i)*. Se non lo avessi fatto, il mio cast esplicito avrebbe riguardato solo sulla costante *'a'*, perché l'operatore di casting ha la precedenza sull'addizione. Il risultato sarebbe stato un'inutile "conversione" da *char* a *char*, e il solito errore di compilazione sull'assegnamento.

MANTIENITI IN ESERCIZIO

Per questo mese hai imparato abbastanza. E' il momento di cimentarti con qualche esercizio.

Esercizio 3: *Scrivi un programma che:*

1) **riempie un array di char di nome *alfabeto* con le lettere dell'alfabeto;**

2) **crea un nuovo array di nome *otebafla*;**

3) **riempie *otebafla* con il contenuto di *alfabeto* al contrario (da *'z'* ad *'a'*);**

☑ ARRAY IN QUATTRO E QUATTR'OTTO

Di solito inizializzare un array è un'operazione piuttosto noiosa:

```
int[] a = new int[3];
a[0] = 12;
a[1] = 2 + 2;
a[2] = 137;
```

Java però ha anche una sintassi speciale per rendere il compito più facile.

```
int[] a = {12, 2 + 2,
                  137};
```

Grazie alle parentesi graffe ho detto due cose in un colpo solo: che l'array ha tre elementi, e quali sono i valori iniziali di questi elementi. Puoi usare questa sintassi solo nel punto in cui dichiari l'array, quindi non puoi usare le parentesi graffe per assegnare nuovi valori ad un array già inizializzato.

☑ ASCII DEI TEMPI ANDATI

Lo standard Unicode è un'estensione del vecchio standard ASCII, che prevedeva solo 256 possibili caratteri ed era insufficiente quando si prendevano in considerazione lingue diverse dall'inglese.

4) **stampa sullo schermo il contenuto di *otebafla*.**

Esercizio 4: *Scrivi un programma che:*

1) **riempie un array di *char* di nome alfabeto con le lettere dell'alfabeto;**

2) **riempie un array di *int* di nome *numeri* con i numeri da 110 a 120;**

3) **crea un nuovo array di *char* di nome *moltiCaratteri*, grande come *alfabeto* e *numeri* messi insieme;**

3) **riempie *moltiCaratteri* con il contenuto di *alfabeto* seguito da quello di *numeri* (il contenuto di numeri deve essere convertito in caratteri);**

4) **stampa sullo schermo il contenuto di *moltiCaratteri*.**

Se riesci a risolvere questi esercizi (o se ne hai capito la soluzione dopo aver sbirciato sul CD) puoi affrontarne uno un po' più complicato:

Esercizio 5: *Scrivi un programma che:*

1) **riempie un array di nome *alfabeto* con le lettere dell'alfabeto (e questo ormai lo sai fare);**

2) **crea un nuovo array di nome *qualcheLettera* composto da soli 10 elementi;**

3) **seleziona dieci lettere a caso da *alfabeto* (anche ripetute) e le usa per riempire *qualcheLettera*;**

4) **stampa sullo schermo il contenuto di *qualcheLettera*.**

Qui incontri per la prima volta il concetto di casualità, che fino ad ora non avevamo mai tirato in ballo. Il modo più semplice per generare numeri casuali in Java è usare la funzione *Math.random()*. Le funzioni sono un argomento piuttosto complesso, che per ora non abbiamo trattato (ne parleremo abbondantemente nei prossimi due mesi, quando cominceremo ad affrontare la programmazione orientata agli oggetti). Le uniche funzioni che hai visto finora sono quelle che stampano sullo schermo: *System.out.println()* e la sua gemella che non va a capo, *System.out.print()*.
La funzione *Math.random()* è un po' diversa: ogni volta che scrivi *Math.random()* il sistema calcola un numero casuale, o come dicono i più precisini "pseudocasuale", compreso tra 0.0 (incluso) e 1.0 (escluso). Il numero viene restituito sotto forma di *double*. Ad esempio, prova a far girare questo programmino:

```
class Random {
  public static void main(String[] args){
    for(int i = 0; i < 4; i++){
      double d = Math.random();
      System.out.print(d + " "); } }
}
```

Una delle prove che ho fatto io ha generato questo output:

0.0035210922116278853 0.02310674969913329 0.9117132792043209 0.83098815566099

Prova a risolvere l'esercizio 5 usando *Math.random()* per generare degli indici casuali per il primo array. Ricorda che *Math.random()* non genera mai il valore 1.0. Ecco il cuore della mia soluzione:

```
char[] qualcheLettera = new char[10];
for(int i = 0; i < qualcheLettera.length; i++){
  // crea un indice random compreso tra 0 e
                                  alfabeto.length-1
  int indiceRandom = (int)(Math.random()
                                * alfabeto.length);
  // inserisci nell'array qualcheLettera un elemento scelto
  // a caso dall'array alfabeto
  qualcheLettera[i] = alfabeto[indiceRandom];
}
```

Concludiamo con un esercizio un po' più interessante:

Esercizio 6: *Scrivi un "generatore di parole casuali" che stampa dieci parole casuali sullo schermo. Ogni parola deve essere composta da 8 lettere, in questo ordine: una lettera qualsiasi, una "sillaba", una seconda sillaba, una vocale. Una sillaba deve essere composta da una vocale seguita da due lettere qualsiasi. Sia le vocali che le lettere qualsiasi devono essere scelte a caso.*

Questo è un esercizio interessante, perché dovrai creare due vettori: uno che contiene tutte le lettere dell'alfabeto, e uno che contiene solo le vocali. Puoi trovare la mia soluzione sul CD (si chiama *ParoleParoleParole.java*). Prova comunque a risolverlo da solo, e non scoraggiarti se ti ci vorrà un po' di tempo. Ne sarà valsa la pena: non capita tutti i giorni di sviluppare un utile programmino che genera parole interessanti come *"wakjurde"*, *"uazoodao"* o *"caskulti"*. Buon divertimento, e arrivederci al mese prossimo!

Paolo Perrotta

☑ UN CICLO DENTRO UN CICLO

Nella mia soluzione all'ultimo esercizio (che trovi solo sul CD) ho usato due cicli "annidati", cioè un ciclo all'interno di un altro ciclo. Volevo stampare due "sillabe", ciascuna composta da una vocale seguita da due lettere qualsiasi. Quindi ho scritto due cicli che vanno da 1 a 2: il più esterno genera un'intera sillaba, il più interno genera una lettera qualsiasi:

```
for(int i = 1; i <= 2;
                  i++){
  <stampa una vocale>
  for(int j = 1; j <= 2;
                    j++)
    <stampa una lettera
              qualsiasi>
}
```

Nei cicli *for* si usa spesso una variabile di controllo di nome i, che sta per "indice". Se servono altre variabili di controllo (ad esempio per costruire più cicli annidati) si usano di solito i nomi j e k.

Presentare informazioni in un elenco "elegante"

La gestione del controllo ListView

Il controllo ListView consente di presentare all'utente informazioni sotto forma di elenco, utilizzando le immagini memorizzate nel controllo ImageList, per renderle più esplicite.

In questo articolo presenteremo il controllo, certamente più utilizzato per mostrae all'utente informazioni sotto forma di elenco: il *ListView*. Le informazioni presentate nel controllo, generalmente presentano, di fianco, un'icona che può essere memorizzata in un controllo *ImageList*. Infine, come di consueto, descriveremo una semplice applicazione per gestire una videoteca in modo da chiarire i concetti presentati nell'articolo.

IL CONTROLLO IMAGELIST

Il controllo *ImageList* (elenco immagini), contiene un insieme d'immagini visualizzabili da tutti i controlli di VB .NET che espongono la proprietà *ImageList*. Un elenco immagini consente di ridurre i tempi di sviluppo, poiché è possibile scrivere codice che fa riferimento ad un unico catalogo di immagini; si tratta di un controllo "invisibile" in fase di esecuzione, perciò, per visualizzare le immagini che esso contiene, deve essere associato ad altri controlli, lo potremo definire un controllo di servizio per altri controlli. Per aggiungere immagini in fase di progettazione si deve:

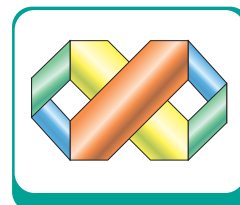
Cliccare sul pulsante con i tre puntini accanto alla proprietà (collezione) *Images*, in questo modo si visualizza la finestra *Editor dell'insieme Image*, il cui semplice utilizzo permette di popolare il controllo.
Nella finestra *Editor* dell'insieme *Image* si deve cliccare sul pulsante *Aggiungi*, per selezionare, dalla finestra di dialogo, le immagini da aggiungere al controllo. Utilizzando le freccette si può variare la posizione e quindi l'indice delle immagini. Con il pulsante *Rimuovi* si elimina un'immagine dall'elenco.

Rispetto a VB6 non è più possibile assegnare una chiave di tipo stringa all'immagine, perciò, per farvi riferimento, si dovrà utilizzare soltanto il relativo indice numerico. In pratica espone una collezione di oggetti *Image*, contenenti ciascuno un'immagine, la collezione *Images* (sostituisce la collezione *ListImages* di VB6), essa deriva dall'interfaccia di base *Ilist,* perciò si possono utilizzare gli usuali *Add* e *Remove* rispettivamente per aggiungere o eliminare immagini da codice in fase di progettazione, si può fare riferimento ad ogni oggetto *Image* utilizzando l'indice numerico nel metodo *Item*. Per aggiungere immagini a livello di programmazione, si deve tener presente che il metodo *Add* accetta oggetti di tipo *Image*, per questo si può utilizzare il metodo *FromFile* che crea un oggetto *Image* dal file specificato, pertanto, per inserire nel controllo un'immagine di esempio si può scrivere:

```
'si deve scrivere l'intero path valido del file
Dim myImage As System.Drawing.Image =
                    Image.FromFile("C:\Esempio.gif")
ImageList1.Images.Add(myImage)
```

Le proprietà peculiari del controllo sono:

La proprietà **TransparentColor** (che sostituisce **MaskColor).** Serve per definire il colore che deve diventare trasparente quando si sovrappone un'immagine a un'altra.
La proprietà **ColorDepth.** Consente di definire il numero di colori da utilizzare per riprodurre le immagini (4, 8, 16, 24 o 32 bit)
La proprietà **ImageSize.** Consente di definire le dimensioni dell'immagine all'interno del controllo. Le eventuali immagini più grandi sono adattate in scala.

Per associare l'elenco immagini ad un controllo, si deve impostare la proprietà *ImageList* del controllo in base al nome del componente *ImageList* (come vedremo in seguito). In pratica, se si associa un controllo *ImageList* ad un altro controllo, non sarà

☑ COLORDEPTH

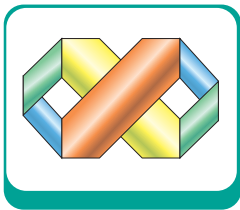
Possibili valori dell'Enumerazione ColorDepth che permette di specificare il numero di colori utilizzati per visualizzare un'immagine in un controllo ImageList.

Depth4Bit - definisce un'immagine a 4 bit

Depth8Bit - definisce un'immagine a 8 bit

Depth16Bit - definisce un'immagine a 16 bit

Depth24Bit - definisce un'immagine a 24 bit

Depth32Bit - definisce un'immagine a 32 bit

necessario scrivere nessuna riga di codice poiché tutto avviene automaticamente. Per l'applicazione di esempio, che descriveremo nel proseguo dell'articolo, popoliamo il controllo con le icone corrispondenti al semaforo verde, rosso e giallo che di norma si trovano nella directory di installazione di Visual Studio .NET nella cartella *\Common7\Graphics\icons\Traffic*.

IL CONTROLLO LISTVIEW

Il controllo *ListView (Controllo Visualizzazione Elenco)*, è uno dei controlli più interessanti ed utili. Per avere subito un esempio di utilizzo del controllo *ListView* in Windows, è sufficiente aprire la finestra *Esplora risorse* in cui la parte destra dell'elenco dei file non è altro che un *ListView*. Il controllo espone due rilevanti oggetti collezione:

- **La collezione *Items* di oggetti *ListViewItem* che rappresenta ogni elemento visualizzato.**
- **La collezione *Columns* di oggetti *ColumnHeader* che rappresenta ogni colonna con la relativa intestazione.**

È disponibile, inoltre, la collezione *SelectedItems* che contiene gli elementi selezionati nel controllo. Il controllo *ListView* consente di mostrare i dati, in esso contenuto, attraverso quattro modalità di visualizzazione, semplicemente modificando la proprietà *View* con una delle seguenti costanti:

LargeIcon modo icone grandi - Permette di visualizzare icone grandi sopra al testo dell'elemento. Gli elementi sono disposti in più colonne (se le dimensioni del controllo lo consentono)
SmallIcon modo icone piccole - Permette di visualizzare icone piccole accanto al testo dell'elemento. Gli elementi sono disposti in più colonne (se le dimensioni del controllo lo consentono)
List modo elenco - Permette di visualizzare icone piccole accanto al testo dell'elemento. Gli elementi sono disposti in un'unica colonna.
Details modo dettaglio - Permette di visualizzare gli elementi in più colonne. Soltanto in questa modalità saranno visualizzate le colonne con le relative intestazioni.

In tutte le modalità è possibile visualizzare le immagini contenute in un controllo *ImageList*, in particolare possono essere visualizzate immagini incluse in tre controlli *ImageList* diversi.

La proprietà LargeImageList consente di associare al *ListView* un controllo *ImageList* da utilizzare per visualizzare immagini in modalità *Icone grandi*.

La proprietà SmallImageList consente di associare al *ListView* un controllo *ImageList* da utilizzare per visualizzare immagini in modalità *List, Details* e *SmallIcon*.
La proprietà StateImageList, consente di associare al *ListView* un controllo *ImageList*, da utilizzare per visualizzare immagini che rappresentano lo stato di un elemento del *ListView* (ad esempio lo stato di selezionato nella modalità *CheckBoxes* pari a *True*). Le immagini di stato vengono visualizzate a sinistra dell'icona posta accanto ad ogni elemento, sono visibili in tutte le visualizzazioni disponibili del controllo *ListView*. Per visualizzare le immagini si deve:

Impostare la proprietà appropriata (*SmallImageList, LargeImageList* o *StateImageList*) sul controllo *ImageList* che si ha intenzione di utilizzare. In fase di progettazione è sufficiente visualizzare la finestra delle proprietà e selezionare dall'elenco a discesa, accanto alle proprietà corrispondenti, uno tra i controlli *ImageList* presenti nella form. Da codice si può scrivere ad esempio:

```
ListView1.LargeImageList = ImageList1
```

Impostare la proprietà *ImageIndex* (o *StateImageIndex*) di ciascun elemento del *ListView* con l'icona associata. In fase di progettazione, si deve visualizzare l'editor dell'insieme *ListViewItem* cliccando sul pulsante con i tre puntini di sospensione accanto alla proprietà *Items* nella finestra delle proprietà, e selezionare un'icona (tra quelle presenti nell'*ImageList* associato) dalla proprietà *ImageIndex*.
Da codice si può scrivere:

```
ListView1.Items(0).ImageIndex = 1
```

Analizziamo ora alcune delle proprietà peculiari del controllo:

AutoArrange permette di specificare se le icone vengono disposte automaticamente e bloccate sulla griglia. Questa proprietà ha effetto soltanto nelle modalità *LargeIcon* e *SmallIcon*.
AllowColumnReorder in modalità *Details*, se impostata a *True*, permette all'utente di modificare l'ordine delle colonne, trascinando le rispettive intestazioni;
CheckBoxes permette di visualizzare, all'interno del *ListView*, le caselle *CheckBox*, in modo da poter selezionare con il segno di spunta alcune righe;
FullRowSelect per dare la possibilità di selezionare l'intera riga (solo in modalità *Details*);
GridLines per visualizzare una griglia tra le righe e le colonne contenenti gli elementi nel *ListView* (in visualizzazione *Details*);
Activation per indicare se un elemento del *ListView* può essere attivato con uno oppure due clic del mouse;

☑ IMAGELIST
Il controllo ImageList può essere utilizzato con i controlli che espongono la proprietà ImageList oppure, nel caso del controllo ListView, delle proprietà SmallImageList e LargeImageList. In particolare può essere associato ai controlli ListView, TreeView, ToolBar, TabControl, Button, CheckBox, RadioButton e Label

☑ IMAGESIZE
La proprietà ImageSize permette di impostare la dimensione delle immagini nell'ImageList agendo sull"oggetto Size che rappresenta le dimensioni di un'area rettangolare definendo altezza e larghezza delle immagini nell'elenco. L'altezza e la larghezza predefinite sono 16x16. La dimensione massima è 256x256.

HoverSelection per consentire di selezionare, automaticamente, un elemento del *ListView* quando il puntatore del mouse viene posizionato per alcuni secondi su di esso;
MultiSelect consente la selezione multipla di più elementi del *ListView.*

LE COLONNE DEL LISTVIEW

Per creare in fase di progettazione le colonne (oggetti *ColumnHeader*) del ListView, si deve cliccare sul pulsante con i tre puntini accanto alla proprietà (collezione) *Columns*, in modo da visualizzare la finestra *Editor* dell'insieme *ColumnHeader*. Premere il pulsante *Aggiungi* e digitare i valori della proprietà:

Text; - consente di inserire il testo visualizzato nell'intestazione delle colonne;
TextAlign; - consente di specificare l'allineamento degli elementi della colonna, i possibili valori sono:

- **Left - allineamento a sinistra;**
- **Right - allineamento a destra;**
- **Center - allineamento centrato;**
- **Width; - larghezza della colonna.**

Sarà possibile referenziare le colonne appena inserite, tramite l'indice numerico visualizzato a sinistra del nome della colonna.

INTESTAZIONE DELLE COLONNE IN FASE D'ESECUZIONE

La creazione ed intestazione delle colonne può avvenire anche in fase d'esecuzione. Ad esempio definiamo la procedura *IntestaColonne*, per popolare la collezione *Column*. In essa dichiariamo la variabile *colX* di tipo *ColumnHeader*.

```
Private Sub IntestaColonne()
  'dichiarazione della variabile di tipo ColumnHeader
  Dim colX As ColumnHeader
  'L'istruzione With consente di eseguire una serie di
            istruzioni su un oggetto specificato senza
                    riqualificare il nome dell'oggetto
  With ListView1.Columns
    colX = .Add("Titolo", 100, HorizontalAlignment.Left)
    colX = .Add("Regista", 100, HorizontalAlignment.Left)
    colX = .Add("Durata", 100, HorizontalAlignment.Left)
  End With
End Sub
```

Per aggiungere un elemento alla collezione *Column* si deve utilizzare il metodo *Add*:

```
Add(testo, larghezza, allineamento)
```

Ognuna di queste proprietà può essere impostata anche separatamente, ad esempio con le seguenti linee di codice:

```
colX.TextAlign = HorizontalAlignment.Right
colX.Width = ListView1.Width \ 3
```

naturalmente queste istruzioni devono essere ripetute per ogni oggetto *ColumnHeader*.

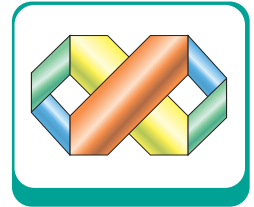
GLI OGGETTI LISTVIEWITEM

Dopo aver definito l'aspetto del *ListView* la cosa più importante è riempirlo di dati, per mezzo dell'oggetto *ListViewItem*. Per creare nuovi elementi del *ListView* in fase di progettazione si deve cliccare sul pulsante con i tre puntini accanto alla proprietà (collezione) *Items* in modo da visualizzare la finestra *Editor* dell'insieme *ListViewItem*. Premere il pulsante *Aggiungi* e digitare i valori delle proprietà:

- **Text contiene il testo presente nel ListView;**
- **Font consente di selezionare il tipo di carattere visualizzato nel ListView;**
- **BackColor e ForeColor consentono di definire i colori di sfondo e del testo di ogni singolo elemento.**

Per aggiungere sottoelementi è sufficiente cliccare sui tre puntini a destra della proprietà (collezione) *SubItems*. Anche in questo caso è possibile modificare l'aspetto di ogni singolo sottoelemento. Un oggetto *ListViewItem* è costituito da testo, dall'indice di un'icona (*ImageIndex*) associata tramite il controllo *ImageList* e, nella visualizzazione *Details*, da una collezione di oggetti *ListViewSubItem* che rappresentano i sottoelementi. Per aggiungere nuovi elementi si deve utilizzare il metodo *Add* della collezione *ListViewItems*

```
ListView1.Items.Add("prova")
```

Per creare un oggetto *ListViewSubItem*, e popolare il ListView nella visualizzazione *Details*, è necessario utilizzare il classico metodo *Add*. Naturalmente anche per i *SubItem* vale la regola che il numero di oggetti *ColumnHeader* determina il numero di oggetti *ListViewSubItem* che è possibile impostare per l'oggetto ListItem.
In particolare il numero di oggetti *ListViewSubItem* è sempre inferiore di uno rispetto al numero di oggetti *ColumnHeader*, poiché il primo oggetto *ColumnHeader* è sempre associato alla proprietà *Text* dell'oggetto *ListViewItem*.

☑ ACTIVATION

La proprietà Activation permette di selezionare il tipo di azione che l'utente deve eseguire per attivare un elemento nel ListView. Le possibili opzioni sono: Standard, OneClick e TwoClick. L'opzione OneClick richiede un solo clic del mouse per attivare l'elemento. L'opzione TwoClick, richiede che l'utente faccia doppio clic per attivare l'elemento, mentre un solo clic modifica il colore del testo dell'elemento. L'opzione Standard richiede che l'utente faccia doppio clic per attivare l'elemento, senza modificarne l'aspetto.

☑ CHECKBOXES

Impostando la proprietà CheckBoxes a True e specificando un controllo ImageList per la proprietà StateImageList, le immagini con indice 0 e 1 vengono visualizzate rispettivamente al posto della casella di controllo non selezionata e della casella di controllo selezionata.

☑ ORDINARE GLI ELEMENTI DEL LISTVIEW

E' possibile attivare la proprietà Sorting per ordinare automaticamente gli elementi presenti nel ListView impostando il tipo di ordinamento in ascendente o discendente. Prestate attenzione, l'ordinamento del ListView è di tipo alfanumerico, per questo i campi numerici e le date non sono ordinati correttamente. L'unico modo per riuscire ad ordinare i numeri e le date, è di formattarli opportunamente. I numeri devono essere formattati come stringhe aggiungendo degli spazi, le date devono essere invece formattate nel formato americano. E' possibile ordinare il contenuto del controllo applicando diversi criteri. Basta semplicemente definire una classe che implementi l'interfaccia IComparer, assegnare un'istanza di tale classe alla proprietà ListViewItemSorter del controllo ListView e invocare, quindi, il metodo Sort. La funzione CompareTo di questa classe riceve un riferimento ai due oggetti ListViewItem che vengono confrontati, pertanto è possibile accedere alle relative proprietà ed a quelle dei corrispondenti sottoelementi.

UNA SEMPLICE VIDEOTECA

Realizziamo una semplice applicazione che mostri l'elenco dei film della nostra videoteca con accanto un semaforo che consigli l'età adatta per la visione. Per il nostro scopo avremo bisogno di:

- **Tre *TextBox* che rappresentino il titolo, il regista e la durata del film con i seguenti nomi: *TextBoxTitolo, TextBoxRegista, TextBoxDurata***
- **Tre *RadioButton* che consiglino l'età adatta per la visione del film, con i seguenti nomi: *RadioButtonTutti, RadioButtonGenitori, RadioButtonAdulti***
- **Un *ListView*, in modalità *Details*, che visualizzi la lista dei film. Richiamando, nell'evento *Load* della form, la procedura *IntestaColonne* vista in precedenza, si otterranno le colonne necessarie.**
- **Un'*ImageList* popolato dalle icone con il semaforo (nell'ordine) verde, giallo e rosso.**
- **Due *Button* per inserire o rimuovere un film dalla lista.**

L'utente dovrà scrivere le caratteristiche del film nei *TextBox* corrispondenti, selezionare un *RadioButton* e premere il bottone *Aggiungi* per inserire il film nella lista. Il codice necessario alla gestione dell'inserimento di un film nella lista sarà, quindi, scritto nell'evento *Click* del button *ButtonAggiungi*.

```
Private Sub ButtonAggiungi_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                     Handles ButtonAggiungi.Click
  Dim Litem As ListViewItem
  Dim SItem As ListViewItem.ListViewSubItem
  Dim Titolo As String; Dim Regista As String
  Dim Durata As Integer
  Titolo = TextBoxTitolo.Text
  Regista = TextBoxRegista.Text
  Durata = TextBoxDurata.Text
  'questo ListItem andrà sotto ColumnHeader(1)
  Litem = ListView1.Items.Add(Titolo)
  Litem.ImageIndex = IndiceIconaSelezionata()
  Litem.ForeColor = Color.Blue
  SItem = Litem.SubItems.Add(Regista)
  SItem.ForeColor = Color.Blue
  SItem = Litem.SubItems.Add(Durata)
  SItem.ForeColor = Color.Red
End Sub
```

Si sono dichiarate le variabili *Litem* di tipo *ListViewItem* ed *SItem* di tipo *ListViewSubItem*, oltre alle variabili che conterranno i valori dei *TextBox*.
L'istruzione:

```
Litem = ListView1.Items.Add(Titolo)
```

aggiunge un nuovo *ListViewItem* nella collezione *Items*, ed ha come effetto di visualizzare il testo contenuto (il titolo del film) in un nuovo rigo del *ListView* (sotto la prima colonna). Per visualizzare l'icona corrispondente alla visibilità del film si assegna alla proprietà *ImageIndex* il valore dell'indice nell'*ImageList* dell'icona corrispondente al *RadioButton* selezionato, restituito dalla funzione *IndiceIconaSelezionata*. L'istruzione successiva assegna il colore Blu al titolo.
Con l'istruzione:

```
SItem = Litem.SubItems.Add(Regista)
```

si può scrivere il testo nelle successive colonne ognuno con un carattere ed un colore diverso. La funzione *IndiceIconaSelezionata* dovrà semplicemente testare il valore della proprietà *checked* dei *RadioButton*, e nel caso sia pari a *True* assegnare alla funzione l'indice dell'immagine (nell'*ImageList*) corrispondente al *RadioButton:*

```
Private Function IndiceIconaSelezionata() As Integer
    If RadioButtonTutti.Checked = True Then
      'semaforo verde
      IndiceIconaSelezionata = 0
    End If
    If RadioButtonGenitori.Checked = True Then
      'semaforo giallo
      IndiceIconaSelezionata = 1
    End If
    If RadioButtonAdulti.Checked = True Then
      'semaforo rosso
      IndiceIconaSelezionata = 2
    End If
End Function
```

Infine nell'evento *Click* di *ButtonRimuovi* si dovrà scrivere il codice necessario a rimuovere dalla lista il film selezionato, utilizzando la classica istruzione *Remove*.
Per evitare incresciosi errori di run-time introduciamo, inoltre, un controllo sul numero di elementi selezionati, se tale numero è pari a zero mostriamo un messaggio uscendo dall'applicazione

```
Private Sub ButtonRimuovi_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                     Handles ButtonRimuovi.Click
  If ListView1.SelectedItems.Count = 0 Then
     MessageBox.Show("Nessun Film Selezionato")
     Exit Sub
  End If
  'rimuove il primo tra gli elementi selezionati
  ListView1.Items.Remove(
                 ListView1.SelectedItems.Item(0))
End Sub
```

Ing. Luigi Buono

Eredità multipla semplificata con le interfacce

Interfacce nella programmazione .NET

Dopo aver esaminato i concetti legati all'ereditarietà, passiamo all'analisi di un'altra fondamentale caratteristica della programmazione orientata agli oggetti con .NET: le interfacce. Riprese direttamente da Java, le interfacce di C# estremizzano il concetto di classe astratta e consentono una sorta di ereditarietà multipla, in versione semplificata.

Le interfacce sono una completa estremizzazione del concetto di classe astratta. Servono per specificare cosa una classe debba fare, ma non come debba farlo. Sommariamente, è possibile pensare alle interfacce come a delle classi astratte costituite da soli metodi astratti. Ciononostante, il paragone rischia di essere forviante. Le interfacce, infatti, rappresentano anche la via di C# e di .NET all'ereditarietà multipla.

DEFININIZIONE DELLE INTERFACCE

La definizione di un'interfaccia, a grosse linee, ricalca quella di una classe:

```
interface NomeInterfaccia {

  tipo-ritorno nomeMetodo1(lista-argomenti);
  tipo-ritorno nomeMetodo2(lista-argomenti);
  tipo-ritorno nomeMetodo3(lista-argomenti);
  ...

}
```

Rispetto alle classi, è importante osservare le seguenti differenze:

La **parola chiave** utilizzata per la definizione di un'interfaccia è *interface*, anziché *class*.
Tutti i **metodi dichiarati** in un'interfaccia non hanno corpo. Pertanto, sono automaticamente metodi astratti, senza il bisogno di dichiararlo esplicitamente con la parola *abstract*.

Prendiamo in esame una prima semplice interfaccia. Nel corso della lezione precedente è stata sviluppata la classe astratta *Animale*. Applicando alcune piccole variazioni, è possibile mutare la classe in un'interfaccia:

```
interface Animale {

  void faiVerso();
  void dimmiCiboPreferito();

}
```

Come avviene con le classi astratte, non è possibile istanziare direttamente un'interfaccia:

```
Animale a = new Animale(); // Errore!
```

Un'interfaccia esiste solamente per essere implementata da una o più classi.

IMPLEMENTAZIONE DI UN'INTERFACCIA

Implementare un'interfaccia significa realizzare una classe che dia definizione ai metodi da essa richiesti. L'implementazione di un'interfaccia, in C#, sintatticamente è simile alla derivazione da una classe base:

```
class NomeClasse : NomeInterfaccia {
```

```
  // ...
}
```

Ecco un esempio pratico:

```
class Gatto : Animale
 {
  public void faiVerso()
  {
    System.Console.WriteLine("Miaaaaaaaaooo!");
  }
  public void dimmiCiboPreferito()
  {
    System. Console.WriteLine("Pesce!");
  }
}
class Cane : Animale {
  public void faiVerso() {
    System.Console.WriteLine("Bau!");
  }
  public void dimmiCiboPreferito()
  {
    System.Console.WriteLine("Carne!");
  }
}
class Topo : Animale
{
  public void faiVerso()
  {
    System.Console.WriteLine("Squit!");
  }
  public void dimmiCiboPreferito()
  {
    System.Console.WriteLine("Formaggio!");
  }
}
```

Sono state realizzate tre classi che implementano l'interfaccia *Animale*. All'interno di ognuna di esse, è stato necessario dare corpo ai due metodi *faiVerso()* e *dimmiCiboPreferito()*. Non si può prescindere da questo punto. Se non si dà implementazione a tutti i metodi richiesti da un'interfaccia, si riceve un errore di compilazione tipo il seguente:

error CS0535: "NomeClasse" non implementa il membro di interfaccia "NomeInterfaccia.nomeMetodo()".

Ora che sono state realizzate le tre classi *Gatto*, *Cane* e *Topo*, è possibile eseguire un test:

```
class Test
{
  public static void Main()
{
    Animale a1 = new Gatto();
    Animale a2 = new Cane();
    Animale a3 = new Topo();
    System.Console.WriteLine("- Gatto ————————");
    a1.faiVerso();
    a1.dimmiCiboPreferito();
    System.Console.WriteLine("- Cane ————————-");
    a2.faiVerso();
    a2.dimmiCiboPreferito();
    System.Console.WriteLine("- Topo ————————-");
    a3.faiVerso();
    a3.dimmiCiboPreferito();
  }
}
```

L'output è abbastanza ovvio:

- Gatto ————————
Miaaaaaaaaooo!
Pesce!
- Cane ————————-
Bau!
Carne!
- Topo ————————-
Squit!
Formaggio!

Benché non sia possibile istanziare direttamente degli oggetti di tipo *Animale*, è possibile creare dei riferimenti di questo tipo. La situazione ricorda da vicino i temi già discussi dell'ereditarietà.
Su un riferimento di tipo *Animale* è possibile richiamare tutti i metodi richiesti dall'interfaccia *Animale*.
Lo smistamento verso la reale definizione del metodo è automatico e affidato direttamente al runtime di .NET.

☑ EREDITÀ MULTIPLA
Come si è detto più volte, le interfacce permettono una sorta di ereditarietà multipla. Anzitutto, una classe può contemporaneamente estendere un'altra classe ed implementare un'interfaccia.

IMPLEMENTARE PIU' INTERFACCE

Come si è detto più volte, le interfacce permettono una sorta di ereditarietà multipla. Anzitutto, una classe può contemporaneamente estendere un'altra classe ed implementare un'interfaccia:

```
class NomeClasse : ClasseBase, Interfaccia
{
  // ...
}
```

Inoltre, ogni classe può implementare due o più interfacce simultaneamente:

```
class NomeClasse : Interfaccia1, Interfaccia2,
```

```
                                    Interfaccia3, ...
{
  // ...
}
```

In generale, quindi, ogni nuova classe può estendere una classe base ed implementare *n* interfacce:

```
class NomeClasse : ClasseBase, Interfaccia1,
                        Interfaccia2, ..., InterfacciaN
{
  // ...
}
```

Si prendano in esame le interfacce esemplificative *StrumentoMusicale* e *OggettoDiLegno:*

```
interface StrumentoMusicale
{
   void suona();
}
interface OggettoDiLegno
{
   void dimmiTipoLegno();
}
```

Una chitarra è sia uno strumento musicale sia un oggetto di legno. Quindi, si può definire una classe *Chitarra* al seguente modo:

```
class Chitarra : StrumentoMusicale, OggettoDiLegno {
  string tipoLegno;
  public Chitarra(string tipoLegno) {
    this.tipoLegno = tipoLegno;
  }
  // Richiesto da StrumentoMusicale:
  public void suona() {
    System.Console.WriteLine("Do Re Mi Fa Sol La Si");
  }
  // Richiesto da OggettoDiLegno:
  public void dimmiTipoLegno() {
    System.Console.WriteLine(tipoLegno);
  }
}
```

In questo caso, ci si può riferire ad un oggetto *Chitarra* in tre maniere differenti:

Usando una variabile di tipo Chitarra, alla maniera classica. In questo caso, tutti i metodi esposti da *Chitarra* possono sempre essere richiamati.

Usando una variabile di tipo StrumentoMusicale. In questo caso, sono a disposizione esclusivamente i metodi esposti dall'interfaccia *StrumentoMusicale*, cioè il solo metodo *suona()*.

Usando una variabile di tipo OggettoDiLegno. In questo caso, sono a disposizione esclusivamente i metodi esposti dall'interfaccia *OggettoDiLegno*, cioè il solo metodo *dimmiTipoLegno()*.

Si verifichi il funzionamento delle strutture realizzate, mediante il seguente test:

```
class Test {

  public static void Main() {
    // Definisco un oggetto Chitarra usando un
                                          riferimento
    // del medesimo tipo.
    Chitarra c = new Chitarra("Palissandro");
    // Uso normalmente i metodi di Chitarra.
    c.suona();
    c.dimmiTipoLegno();
    // Passo ad un riferimento di tipo
                                   StrumentoMusicale.
    StrumentoMusicale s = c;
    // Ora ho a disposizione solo suona();
    s.suona();
    // Passo ad un riferimento di tipo OggettoDiLegno.
    OggettoDiLegno o = c;
    // Ora ho a disposizione solo dimmiTipoLegno();
    o.dimmiTipoLegno();
  }
}
```

EREDITARIETA' TRA INTERFACCE

Due interfacce possono derivare l'una dall'altra. Il meccanismo è semplice:

```
interface Interfaccia1 {
  void metodo1();
}
interface Interfaccia2 : Interfaccia1 {
  void metodo2();
}
```

Interfaccia2 eredita da *Interfaccia1*. Questo significa che le classi che implementeranno *Interfaccia2* dovranno definire sia i metodi richiesti da *Interfaccia2* sia quelli voluti da *Interfaccia1*.
Ecco un esempio:

```
class Esempio : Interfaccia2 {
  // Richiesto da Interfaccia1.
  public void metodo1()
```

☑ CLASSI ASTRATTE E METODI ASTRATTI

Le classi astratte sono classi create appositamente per essere derivate. Una classe astratta può essere derivata, ma non è possibile avere delle sue istanze da impiegare direttamente in un software. Alla base dell'astrazione di una classe è la parola chiave abstract

```
{
   System.Console.WriteLine("metodo1()");
 }
 // Richiesto da Interfaccia2.
 public void metodo2()
 {
   System.Console.WriteLine("metodo2()");
 }
}
```

Agli oggetti di tipo *Esempio* ci si potrà riferire sia usando variabili di tipo *Esempio*, sia quelle di tipo *Interfaccia1*, sia quelle di tipo *Interfaccia2*, secondo le norme già note:

```
class Test {
  public static void Main()
{
   Esempio e = new Esempio();
   e.metodo1();
   e.metodo2();
   Interfaccia1 i1 = e;
   i1.metodo1();
   Interfaccia2 i2 = e;
   i2.metodo2();
 }
}
```

INTERFACCE E CASTING

Si torni a considerare l'esempio di *Animale, Gatto, Cane* e *Topo*. E' stato detto che una variabile di tipo *Animale* (che è un'interfaccia) può tenere riferimento di un oggetto di tipo *Gatto, Cane* o *Topo* (che sono classi che implementano l'interfaccia *Animale*).
La faccenda è semplice:

```
Animale a = new Gatto();
```

In certi casi, si desidera poter compiere l'operazione inversa.
Ad esempio, avendo a disposizione un riferimento di tipo *Animale*, si immagini di voler risalire al riferimento verso la reale classe dell'oggetto.
L'operazione è possibile attraverso un casting esplicito, a patto di conoscere quale classe sia quella di appartenenza dell'oggetto:

```
Animale a = new Gatto();
// ...
Gatto g = (Gatto)a;
```

Con questa tecnica, è possibile tornare al riferimento più dettagliato. Il trucco, oltre che con le interfacce, funziona anche nei casi di semplice ereditarietà.

CONCLUSIONI

Mancano pochi tasselli al completamento dello studio della programmazione orientata agli oggetti con C# e .NET. Rinnovo l'appuntamento per il mese prossimo, per parlare di strutture ed enumerazioni, che sono tra gli aspetti più originali e peculiari di C# e .NET. Vi aspetto!

Carlo Pelliccia

☑ BIBLIOGRAFIA

Guida a C#
Herbert Schildt
McGraw-Hill, 2002
ISBN 88-386-4264-8

Introduzione a C#
Eric Gunnerson
Mondadori Informatica, 2001
ISBN 88-8331-185-X

C# Guida per lo sviluppatore
Simon Robinson e altri
Hoepli, 2001
ISBN 88-203-2962-X

PROGRAMMAZIONE IN VISUAL C# .NET

Il linguaggio C#, il nuovo linguaggio Microsoft e parte integrante del framework .NET si appresta a diventare uno dei linguaggi di punto per la programmazione del nuovo millennio. In questo testo l'autore nei primi capitoli mostra come realizzare un semplice servizio Web e come creare un'applicazione in grado di sfruttarlo, rendendo così familiari al programmatore le risorse a sua disposizione; nel prosieguo si addentra nei dettagli di C# e della sua relazione con il Framework .NET, in modo che il lettore si ritroverà in grado, al termine del volume, di affrontare i più svariati aspetti di programmazione. Sono proposti svariati argomenti, tra questi:

- **i file e gli oggetti di serializzazione;**
- **la creazione di programmi di messaggistica;**
- **l'utilizzo di XML e ADO .NET per interagire con i database.**

- **Difficoltà: Medio – Alta**
- **Autore: Harold Davis**
- **Editore: MCGraw-Hill**
  http://www.informatica.mcgraw-hill.it
- **ISBN: 88-386-4311-3**
- **Anno di pubblicazione: 2003**
- **Lingua: Italiano**
- **Pagine: 624**
- **Prezzo: € 38,00**

Controllare gli eventi "anomali" nelle applicazioni

La gestione delle eccezioni

Una caratteristica conosciuta da pochi è la possibilità di gestire eventi anomali di un programma tramite il meccanismo delle eccezioni. Vediamo come funziona.

Molte volte si è parlato su queste pagine di *robustezza del codice*, cioè di quella caratteristica che misura la capacità del software di "reagire", in maniera ragionevole, al presentarsi di situazioni anomale. Ad esempio un programma che, non trovando un file che deve aprire, mostra un messaggio di errore, è più robusto di un programma che nelle stesse condizioni causa un crash del sistema. Inutile dire che più un software è robusto, maggiore è la sua qualità. Una buona strada da percorrere per rendere il proprio codice robusto, è quella di cercare di prevedere tutti i possibili errori che possono presentarsi in una certa situazione e scrivere del codice adatto a trattarli. Purtroppo questo è un metodo di programmazione usato da pochi. Quando programmiamo, infatti, siamo naturalmente portati all'ottimismo (che sarà pure il sale della vita, ma si sa che troppo sale fa male...) e quindi tendiamo a scrivere del codice che funzionerà soltanto nel migliore dei modi possibili, che, come ogni programmatore sa, è quello in cui non si verificano errori software. Forse questa tendenza è dovuta al fatto che scrivere codice per la gestione di errori è una cosa noiosa e ripetitiva, inoltre rende il più delle volte il codice stesso difficile da leggere e capire. La cosa ideale sarebbe avere un meccanismo che renda possibile *separare* la parte di codice "interessante" (quella del mondo perfetto) dalla parte, noiosa ma necessaria, della gestione degli errori. Ovviamente i progettisti del C++ non fanno mancare nulla e hanno implementato questa caratteristica col meccanismo delle *eccezioni*.

ECCEZIONALE VERAMENTE

Per "eccezione" si intende per l'appunto "situazione anomala", in altre parole il fatto che qualcosa è andato storto. Una eccezione può verificarsi virtualmente in qualsiasi parte del codice e, ovviamente, scrivere un blocco di istruzioni di trattamento dell'errore per ogni singola linea di codice del programma sarebbe una strada impercorribile. L'idea quindi è quella di raggruppare una serie di istruzioni e dire al programma: "Prova ad eseguirle, se qualcosa va storto esegui quest'altro codice".
Le parole chiave del C++ che riguardano questo meccanismo sono *try* e *catch*; il loro utilizzo è il seguente:

```
try
{
//blocco istruzioni senza trattamento dell'errore
}
catch (<tipo di eccezione> <identificatore>)
{
//gestione dell'errore
}
```

Il blocco di istruzioni preceduto dalla clausola *try* è un blocco istruzioni ordinario del C++, quindi per esso valgono tutte le regole di visibilità delle variabili viste nelle prime puntate di questo corso. Per quanto riguarda il blocco preceduto dalla parola chiave *catch*, è da notare il fatto che in esso è possibile definire un particolare *<identificatore>*, che rappresenterà l'oggetto "eccezione" all'interno del blocco.
Questo oggetto è di tipo *<tipo eccezione>* e può essere, ad esempio, di uno qualsiasi dei tipi predefiniti del C++ ma, più verosimilmente, sarà una istanza di una classe creata apposta per contenere informazioni relative al tipo di errore che si è verificato. Parleremo in seguito delle caratteristiche di questo blocco, dopo avere analizzato in che modo sia possibile, come si dice in gergo, *"lanciare" un'eccezione.*

LANCIALA E VAI A RIPRENDERLA

Quando scriviamo codice capita, a volte, di avere a che fare con errori che nel contesto specifico non sappiamo come trattare e che desidereremmo fossero trattati a un livello più alto del software. Quello che dobbiamo fare in questi casi, è racchiudere tutte le informazioni, relative all'errore, in un oggetto e lanciare tale oggetto come eccezione. Questa operazione viene compiuta in C++ mediante la parola chiave *throw*. Supponiamo ad esempio di dovere implementare una funzione per richiedere da consolle un numero intero positivo. Il corpo di tale funzione potrebbe essere il seguente:

```
int RichiediInteroPositivo()
{
   int numero;
   cout << "Ciao! Inserisci un intero positivo: ";
   cin >> numero;
   return numero;
}
```

Ovviamente, come è facile vedere, la funzione è incompleta, in quanto manca un controllo sul valore intero inserito, qualcosa del tipo:

```
if (numero < 0) { /* ERRORE! */}
```

Il problema di inserire questa istruzione, consiste nel fatto che non sappiamo, in questo contesto, come tale errore debba essere trattato; in altre parole non conosciamo quale debba essere il contenuto del blocco istruzioni dell'*if*. Quello che allora dobbiamo fare è costruire un oggetto apposito, in grado di contenere l'informazione riguardante l'errore avvenuto. Questo oggetto potrebbe essere una istanza della seguente, semplice, classe:

```
class ErroreInteroNegativo
{
public:
   ErroreInteroNegativo(int num) : numero(num) {}
   int GetValore() {return numero;}
private:
   int numero;
};
```

Questa classe porta con se l'informazione più elementare che potremmo pensare di trasmettere, cioè il valore negativo inserito. A questo punto siamo pronti a inserire nel codice della funzione *RichiediInteroPositivo()* il controllo di validità sul dato inserito; esso apparirà più o meno così:

```
int RichiediInteroPositivo()
{
   int numero;
   cout << "Ciao! Inserisci un intero positivo: ";
   cin >> numero;
   if (numero < 0) throw ErroreInteroNegativo(numero);
   return numero;
}
```

L'uso di *try* e *catch* per la chiamata di *RichiediInteroPositivo()* prevede la scrittura di codice simile al seguente:

```
try
{
   //... codice
   RichiediInteroPositivo();
   //... altro codice
}
catch (ErroreInteroNegativo ein)
{
   //gestione dell'errore
   cout << "Attenzione! Hai inserito un numero negativo!\n";
}
```

Come si può vedere, il blocco istruzioni *catch* gestisce l'errore semplicemente stampando un messaggio a schermo. Tuttavia, questo è stato fatto solo a scopo didattico, in realtà la gestione dell'errore potrebbe essere molto più complessa (ad esempio prevedere dei *rollback* su un database, oppure la chiusura di una connessione di rete ecc.); la cosa interessante risiede nel fatto che chi ha scritto la funzione *RichiediInteroPositivo()* non ha dovuto minimamente pensare al modo in cui trattare un eventuale errore, ma ha demandato tutta la fatica a chi ha scritto la relativa clausola *catch*. Inoltre, in questo modo, tutta la parte della gestione dell'errore è concentrato in una ben precisa porzione di codice invece di essere sparsa in giro per il programma. La flessibilità di questo costrutto è estrema, infatti è possibile, ad esempio, utilizzare l'informazione portata dall'oggetto eccezione per raffinare la gestione dell'errore. Supponiamo ad esempio che se il valore inserito è compreso tra -10 e -1 l'utente abbia la possibilità di inserirne uno nuovo, altrimenti il programma debba terminare la sua esecuzione.
Il codice per fare questo è:

```
int num = 0;
while (true)
{
   //ciclo infinito!
   try
   {
      num = RichiediInteroPositivo();
      break; //esce dal "while"
   }
   catch (ErroreInteroNegativo ein)
   {
```

☑ PER SAPERNE DI PIÙ

A chi volesse approfondire la sua conoscenza sulle librerie standard del C++, consigliamo il validissimo libro

• THINKING IN C++ - 2ND ED. - VOLUME 2" Bruce Eckel e Chuck Allison

che rappresenta sicuramente un ottimo riferimento per i programmatori più avanzati (o aspiranti tali) ed è oltretutto disponibile gratuitamente per il download, partendo dall'indirizzo http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html

Se volete invece dare un'occhiata a una reference delle funzioni standard del C per la manipolazione di stringhe (o meglio: di array di caratteri terminati da "\0" :-) consultate l'indirizzo: http://www.cplusplus.com/ref/cstring/

```
    if (ein.GetValore() >= -10)
        continue; //altro ciclo "while"
    else
    {
        cout << "Questo non lo dovevi fare...\n";
        exit(-1); //esco dal programma
  } }
}
cout << "Bravo! Hai inserito il numero *positivo* "
<< num << "!\n";
```

Come si vede l'oggetto eccezione è stato utilizzato all'interno del blocco catch esattamente come una variabile generica, tramite il suo identificatore *ein*. Inoltre non è stato necessario modificare in alcun modo il corpo della funzione *RichiediInteroPositivo()*. Vediamo di seguito una gestione ancora più sofisticata della clausola *catch*.

TRY&CATCH MULTIPLI

All'interno di un blocco di codice possono essere presenti molte fonti di errori, quindi molte fonti di eccezioni. In effetti, la struttura dei blocchi *try&catch* può essere estesa in maniera molto immediata per fare fronte all'esigenza di controllare tutte le possibili eccezioni in un blocco di codice: basta inserire tutte le clausole *catch* necessarie. In altre parole, la struttura vista all'inizio va modificata nel modo seguente:

```
try
{
  //blocco istruzioni
}
catch (<tipo di eccezione 1> <identificatore>)
{
  //gestione dell'errore
}
catch (<tipo di eccezione 2> <identificatore>)
{
  //gestione dell'errore
}
catch (<tipo di eccezione 3> <identificatore>)
{
  //gestione dell'errore}
  //e cosi' via...
```

Come i lettori più attenti avranno notato, per usare un blocco di questo tipo abbiamo bisogno di tante classi quante sono le possibili eccezioni nel blocco di codice: ogni *catch* cattura una sola eccezione, di un solo tipo, e se in un blocco possono venire sollevate *N tipi* di eccezioni allora dovremo avere *N classi* che descrivano ognuna un tipo di eccezione. Vediamo un piccolo esempio, modificando un po' il codice dell'esempio precedente. Innanzi tutto introduciamo qualche nuova classe per le nostre eccezioni:

```
class ErroreZeroDieci
{ //codice della classe };
class ErroreMenoDieci
{ //codice della classe };
```

Con la prima classe verrà identificata l'eccezione sollevata nel caso in cui l'input sia un numero negativo, ma esso sia compreso tra -1 e -10 (in questo caso daremo all'utente un'altra chance di inserimento). La seconda classe serve ad incapsulare un'eccezione data da un input negativo inferiore a -10 (e in questo caso non verrà data altra chance all'utente). Predisporre le classi non è sufficiente: dobbiamo modificare anche il blocco di codice da controllare inserendo la nuova gestione delle eccezioni, ma soprattutto dobbiamo modificare la funzione *RichiediInteroPositivo()* affinchè possa lanciare le eccezioni in maniera corretta. La nuova funzione *RichiediInteroPositivo()* assumerà adesso la seguente forma:

```
int RichiediInteroPositivo()
{
  int num;
  cout << "Ciao! Inserisci un intero positivo: ";
  cin >> num;
  if (num<0 && num>-10) throw ErroreZeroDieci();
  if (num<-10) throw ErroreMenoDieci();
  return num;
}
```

Mentre il blocco di codice da tenere sotto controllo diverrà:

```
int num = 0;
while (true)
{
  //ciclo infinito!
  try
  {
      num = RichiediInteroPositivo();
      break; //esce dal "while"
  }
  catch (ErroreZeroDieci ezd)
  {
     cout << "Pentiti! E reinserisci un nuovo numero...\n";
     continue; }
  catch (ErroreMenoDieci emd)
  { cout << "Cosi' proprio non va...\n";
     exit(-1); }
}
//fine while
cout << "Bravo! Hai inserito " << num;
cout << " che in effetti e' *positivo*!\n";
```

A guardare bene quello che abbiamo fatto e come lo

☑ THROW

L'esecuzione di throw provoca l'uscita dal blocco in cui essa si trova; in questo contesto si dice che la funzione ha sollevato un'eccezione. La throw accetta un argomento come parametro che viene utilizzato per creare un oggetto, che non "ubbidisce" alle classiche regole di scope e che viene restituito a chi ha tentato l'esecuzione dell'operazione.

abbiamo fatto, una cosa è sicuramente evidente: un codice ben strutturato e robusto è qualcosa che nasce da prima del debug. Le eccezioni devono far parte del nostro modo di pensare all'interno di un programma così come il pensare a classi e l'elaborazione di un algoritmo, e questo perché codici ed errori sono legati in maniera indissolubile. Prima di pensare alla parte successiva di codice, bisogna sempre pensare alle possibili eccezioni in quella appena scritta.

L'ACCHIAPPA-TUTTO

Siccome nessuno è perfetto, non possiamo aspettarci di aver pensato a tutto: magari questa volta sì, ma prima o poi succederà che ci dimenticheremo qualcosa, oppure qualche fonte di eccezioni sfuggirà alla nostra vista. Ci sono bug che si annidano nel codice per anni e vengono fuori in combinazioni di eventi incredibilmente improbabili, ma non impossibili. I software real-time di una centrale nucleare o di un razzo spaziale non possono permettersi il lusso di una eccezione non gestita, ad esempio. Sta alla bravura del programmatore usare bene il meccanismo di cattura delle eccezioni. Nel *try&catch* possiamo avvalerci di una clausola generica di cattura di eccezioni, la quale serve per gestire le più impreviste tra le situazioni impreviste (perdonate il gioco di parole).
Ad esempio, in un software di importanza strategica, che sia real-time, potrebbe verificarsi qualche situazione inattesa che generi una eccezione non gestita (magari perché proprio non considerata possibile o perché non ci si è avveduti della sua possibile esistenza): in quel caso dobbiamo poter portare il sistema che fa uso di tale software in uno stato stabile che non pregiudichi la sicurezza di niente o nessuno.
Possiamo pensare ad una struttura di questo tipo:

```
try
{ //blocco istruzioni  }
catch (<tipo di eccezione 1> <identificatore>)
{ //gestione dell'errore }
//e cosi' via...
catch (<tipo di eccezione 2> <identificatore>)
{  //gestione dell'errore  }
catch (...)
{  //gestione dell'errore generico:
   //si porta il sistema in uno stato sicuro }
```

Questa struttura è come quella che abbiamo visto nel paragrafo precedente, solo che abbiamo aggiunto una nuova clausola *catch* il cui argomento non è una convenzione tipografica: il simbolo "..." è parte del vocabolario C++ e sta ad indicare una generica lista di argomenti.

☑ CONTATTA GLI AUTORI!

Se hai suggerimenti, critiche, dubbi o perplessità sugli argomenti trattati e vuoi proporle agli autori puoi scrivere agli indirizzi:

alfredo.marroccelli@ioprogrammo.it **(Alfredo)**
e
marcodelgobbo@ioprogrammo.it **(Marco)**

Questo contribuirà sicuramente a migliorare il lavoro di stesura delle prossime puntate.

LEZIONI DI C++

**Un testo che rende particolarmente agevole l'apprendimento dei fondamenti della programmazione e del codice C++.
L'autore John Smiley, presidente della Smiley and Associates, un'azienda di consulenza informatica, è autore di altri otto libri e, con Lezioni di C++, ha pensato bene di fornire un testo adatto ad un lettore neofita, che non ha nessuna esperienza in campo di programmazione e che comunque si appresta a voler apprendere un linguaggio come il C++ che, sicuramente, non si annovera tra i linguaggi più semplici nell'apprendimento. Gli argomenti, secondo lo stile dell'autore e ormai divenuto un suo standard, sono trattati in modo semplice e "divertente". Tra gli argomenti trattati: imparare i concetti della programmazione applicabili a più linguaggi; sviluppare delle competenze in C++ per il mondo reale e utilizzare la programmazione a oggetti; lavorare con le variabili, le costanti e i dati tipizzati del C++.**

Difficoltà: Medio – Alta
Autore: J.Smiley
Editore: McGraw-Hill
http://www.informatica.mcgraw-hill.it
ISBN: 88-386-4325-3
Anno di pub.: 2003
Lingua: Italiano
Pagine: 573
Prezzo: € 32,00

Quindi, l'ultimo *catch* è quello che ci permette di non far scappare alcuna eccezione: non c'è molto da fare quando si verifica una eccezione non gestita da altri *catch* precedenti (vuol dire che è successo qualcosa di cui si ignorava l'esistenza), tuttavia è mediante quel *catch* che ad esempio si potrebbero attivare le procedure di emergenza di una centrale nucleare (ovviamente, in un software real-time di quel tipo vengono a cadere molte delle nostre "convenzioni" di programmazione, quindi probabilmente le eccezioni non hanno motivo di essere pensate: questo esempio è solo a fini didattici, per dare un'idea).

CONCLUSIONI

Abbiamo appreso un meccanismo molto potente per il controllo degli errori nei nostri programmi: ci salverà (probabilmente) spesso da seri problemi, oltre a farci guadagnare molte ore di sonno.
E' sempre bene trattare le eccezioni nei propri programmi, perché la robustezza è un requisito della qualità.

Alfredo Marroccelli e Marco Del Gobbo

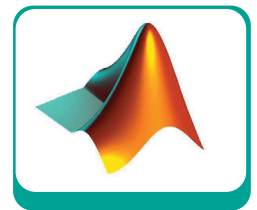
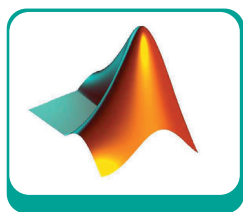
Sviluppare "applicazioni" con MATLAB

Uno sguardo all'ambiente di sviluppo

Dopo aver acquisito una buona conoscenza delle potenzialità di calcolo di MATLAB, rivolgiamo il nostro interesse ad argomenti maggiormente attinenti all'uso dell'ambiente. In altre parole esaminiamo quali siano le caratteristiche che influiscono sulla produttività in misura, per lo meno uguale, alle funzioni del linguaggio di programmazione che abbiamo potuto esplorare nei numeri precedenti.

Nei primi appuntamenti dedicati a MATLAB abbiamo volutamente dato maggiore spazio alla trattazione delle caratteristiche di calcolo del prodotto, poiché é naturale pensare che queste siano quelle caratterizzanti e che maggiormente incuriosiscono coloro i quali abbiano la necessità di disporre di uno strumento di calcolo. Abbiamo volutamente trascurato la trattazione di molte di quelle informazioni che fossero attinenti all'uso dell'ambiente MATLAB. Per citare un buon esempio, che rende chiaro il concetto, proviamo a pensare all'uso di un debugger. Chi abbia sviluppato programmi e algoritmi di una certa complessità, sa bene che un programma può mostrare una complessa casistica di malfunzionamenti; scoprire quale sia la causa di un errore può essere un compito lungo e doloroso. Soprattutto quando ci occupiamo di calcolo, sappiamo bene che una delle situazioni peggiori in cui ci possiamo imbattere è quella in cui un subdolo errore genera un risultato del calcolo non corretto. Quello che rileviamo sono numeri "quasi" esatti che non bloccano il fluire delle informazioni, ma ci sfuggono proprio per la loro apparente esattezza. Alla lunga inquinano sempre più il risultato numerico ma rimangono comunque molto difficili da scoprire. Nel momento in cui ce ne rendiamo conto, ciò che immediatamente ci preoccupa è il processo di ricerca del punto in cui abbiamo generato l'errore. Ed ecco che il debugger ci viene in soccorso poiché ci consente di percorrere il programma un'istruzione alla volta e verificare *al volo* i valori delle variabili. In questo numero vedremo all'opera sia il Debugger sia altre applicazioni di ausilio all'uso di MATLAB interattivo e procedurale.

LA COMMAND HISTORY

Durante l'uso interattivo di MATLAB, abbiamo imparato a conoscere la finestra chiamata *Command Window* il cui uso è molto semplice. Infatti, essa consente di scrivere istruzioni e di visualizzare immediatamente il risultato della loro elaborazione. Non sappiamo ancora che tutto quello che noi scriviamo viene effettivamente registrato e può essere utilizzato in seguito. Una seconda finestra dell'ambiente MATLAB ci mostra la storia delle istruzioni digitate sulla *Command Window*. Da questa finestra (vedi Fig. 1) abbiamo la possibilità di eseguire alcune semplicissime azioni che si rivelano utili e veloci. Cliccando due volte sul singolo comando, che appare nella lista, si forza la sua immediata esecuzione. Mentre, se si clicca col tasto destro appare un menu che ci dice che possiamo copiare il comando o addirittura creare direttamente un m-file che contenga le righe che abbiamo selezionato. Se proviamo ad eseguire quest'opzione vediamo immediatamente comparire le righe evidenziate sull'editor di

☑ LANCIO DELLA APPLICAZIONE

L'applicazione puo' essere lanciata dalla Command Window per mezzo di:

>> workspace

☑ ARRAY EDITOR

L'Array Editor può essere invocato dalla Command Window per mezzo di:

>> open solemese

É necessario specificare il nome di una variabile presente in workspace

```
Command History
hs=surf(italy,'facecolor','interp');
hold on
set(hs,'EdgeColor','none');
max(max(italy))
[c,h0]=contour3(italy,[0 0],'r');
set(h0,'LineWidth',2)
[c,h20]=contour3(italy,[20 20],'y');
set(h20,'LineWidth',2)
minimo=min(min(italy));
massimo=max(max(italy));
[r_topo,c_topo]=size(topomap);
a=find(italy<0);
b=find(italy>=0);
soglia=101;
[r_it,c_it]=size(italy);
cdata=italy*0;
cdata(a)=fix((italy(a)-minimo)/(abs(minimo)/soglia)+1);
cdata(b)=ceil(italy(b)/(massimo/(r_topo-soglia-1))+soglia+1);
cdata=reshape(cdata,r_it,c_it);
set(hs,'cdata',cdata);
colormap(topomap)
light('position',[0 100 10000])
lighting phong
set(gcf,'Renderer','zbuffer')
hold off
[x,y]=ginput;
set(gca,'CameraViewAngleMode','manual');
camva(20)
camproj perspective
daspect([1,1,500]);
```

***Fig. 1:* Command History.**

MATLAB. Abbiamo gia usato l'Editor per scrivere i nostri primi programmi, e nel seguito vedremo alcuni dettagli utili su quest'applicazione così preziosa. Se invece trasciniamo la selezione sulla *Command Window*, ne eseguiamo una semplice copia ma il codice non viene automaticamente eseguito. MATLAB attende il successivo *Enter* per effettuare il calcolo. In altre parole, la *Command History* é un contenitore che ci consente di recuperare dal lavoro interattivo porzioni di codice che possono rivelarsi utili per una loro riesecuzione o per la composizione di programmi e algoritmi. Abbiamo compreso dalle nostre esperienze precedenti, che l'uso interattivo di MATLAB è destinato a qui tentativi di elaborazione dati che non debbano necessariamente divenire parte di un algoritmo. Avviene spesso invece che queste manipolazioni esplorative siano un buon punto di partenza o di riferimento per quello che deve divenire l'algoritmo definitivo. La conservazione del lavoro interattivo del passato si rivela ora particolarmente utile. Una variazione sul tema è la funzione *diary*. Se digitiamo il semplice comando *diary* sulla *Command Window*, attiviamo la registrazione su file della sessione di lavoro fino alla successiva digitazione del comando *diary*. A questo punto viene generato un file di nome *diary* contenente la copia fedele di quello che é apparso sulla *Command Window*, comandi e risultati compresi.
È poi possibile aprire questo file per mezzo dell'*Editor* e modificarlo per estrarne le parti che riteniamo più utili per il nostro lavoro. Un suo uso un po' più sofisticato prevede che si possa specificare un nome di file sul quale riversare la copia della sessione di lavoro.
È sufficiente usare la seguente sintassi:

```
>> diary('nomefile.ext')
```

CURRENT DIRECTORY

Questa finestra ha l'aspetto di una lista di file e permette la navigabilità delle directory in una maniera molto intuitiva. Essa è utile per vedere i file che fanno parte di una directory, potere facilmente aprirli nell'Editor ed eseguire nell'ambiente.

WORKSPACE

La finestra che mostra il contenuto del workspace (vedi Fig. 2) consente alcune funzionalità interattive di una certa comodità. Se proviamo a cliccare doppio su una variabile, causiamo l'apertura di un'applicazione ausiliaria chiamata *Array Editor*.

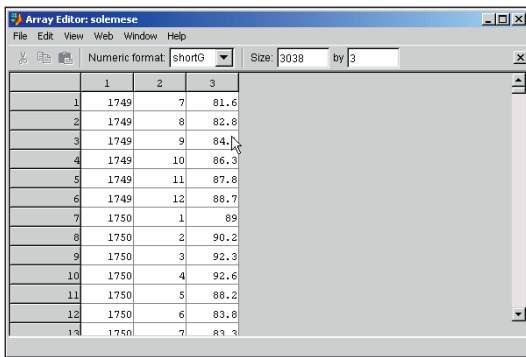
***Fig. 2:* Il WorkSpace.**

La variabile in questione, a questo punto, può essere modificata interattivamente (vedi Fig. 3). Qui abbiamo visualizzato una variabile proveniente da un file che abbiamo imparato a leggere in uno dei numeri precedenti. Esso riporta le date (anni e mesi) e le registrazioni del numero medio di macchie solari nel periodo. Questi dati si trovano nel file *solemese.xls* e sono presenti ora in MATLAB poiché sono stati importati per mezzo dell'*Import Wizard* (vedi Fig. 4).

Fig. 3: ***Array Editor.***

Esso ci consente di importare vari formati di file in maniera interattiva, e ci risparmia l'apprendimento della sintassi di comandi specializzati.

L'*Import Wizard* può essere invocato sia dall'*Array Editor* sia da MATLAB per mezzo del menu *File* e di un item chiamato "*Import Data…*".

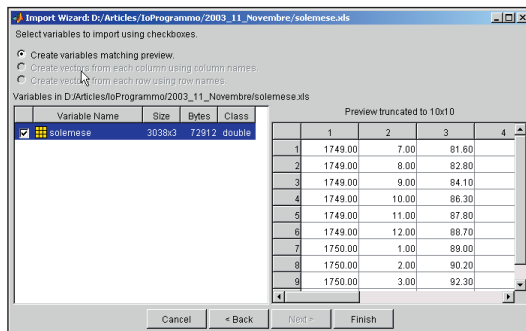



Fig. 4: ***Import Wizard.***

IL LAUNCH PAD

Il *Launch Pad* di MATLAB elenca gli strumenti e le applicazioni di servizio di ogni prodotto.

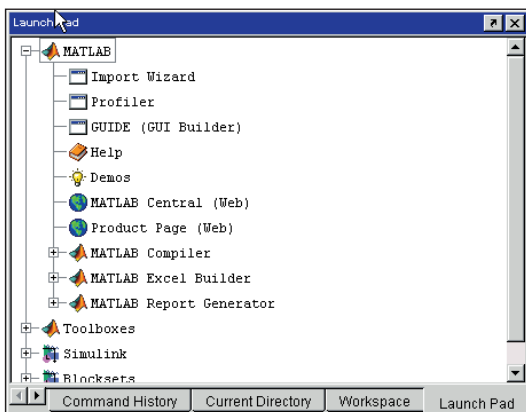



Fig. 5: ***Launch Pad.***

Inoltre, è disponibile il riferimento al sistema di help che contiene tutta la documentazione disponibile circa l'uso dei singoli strumenti e funzioni.

L'EDITOR

Lo scopo principale di un editor é quello di fornire un ambiente comodo dove poter creare un programma. Descritto in questo modo può suonare semplicistico; in realtà ci troviamo di fronte ad un'applicazione di servizio agile ma allo stesso tempo sofisticata.

Se apriamo un programma MATLAB (vedi Fig. 6) e lo visualizziamo, ci accorgiamo di alcune caratteristiche non banali.

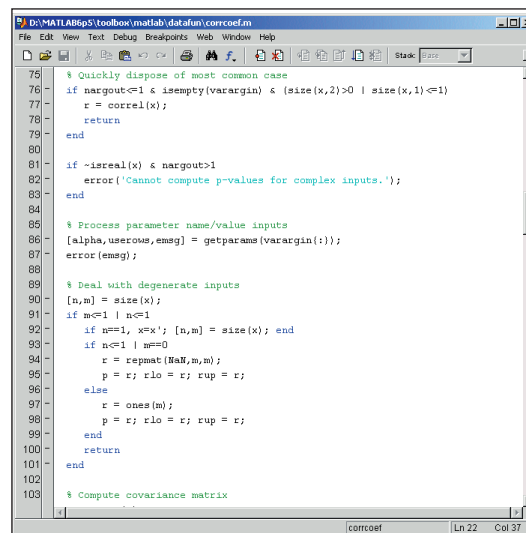
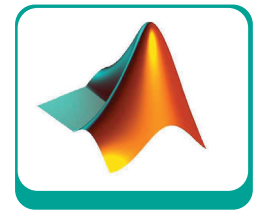



Fig. 6: ***Editor.***

Prima di tutto notiamo che compaiono i numeri che individuano la singola riga all'interno del codice. Questi sono particolarmente utili poichè sono il riferimento principale quando viene segnalato un errore. MATLAB informa l'utilizzatore mostrando il numero di riga dove si è verificato un problema insormontabile. Per esempio:

```
>> corrcoef(0);
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress
                                      this warning.)
> In D:\MATLAB6p5\toolbox\matlab\datafun\
                  corrcoef.m (correl) at line 189
  In D:\MATLAB6p5\toolbox\matlab\datafun\
                              corrcoef.m at line 77
```

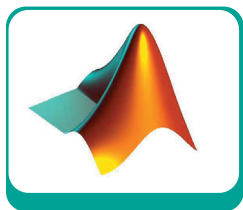
Inoltre, notiamo che l'*Editor* colora in maniera differente alcune parole. Nell'esempio di Fig. 6 vediamo quattro differenti colori: il verde evidenzia i commenti, il blu fa risaltare le funzioni appartenenti al linguaggio MATLAB,

☑ IMPORT WIZARD

L'Import Wizard può essere invocato dalla Command Window per mezzo di:

```
>> uiimport
```

☑ EDITOR/ DEBUGGER

L'Editor/Debugger può essere invocato dalla Command Window per mezzo di:

```
>> edit
```

il celeste è riservato alle stringhe ed infine il nero viene usato per tutti gli altri caratteri (nomi di variabile e separatori). Anche qui è possibile evidenziare alcune linee di codice e farle eseguire nella *Command Window* di MATLAB semplicemente attivando l'appropriato menu, per mezzo del tasto destro.

IL DEBUGGER

In moltissimi casi, in cui si deve individuare il punto in cui si verifica un errore, è di estremo aiuto poter seguire l'elaborazione un passo alla volta. Per fare questo è necessario stabilire un punto preciso da cui iniziare il debug e possedere quindi un meccanismo di avanzamento manuale dell'elaborazione. Tutto questo si ottiene per mezzo dell'uso dei bottoni presenti sulla Toolbar dell'*Editor* MATLAB e riportati in Fig. 7.

***Fig. 7:* Gestione del Debug.**

Diamo uno sguardo ai primi tre bottoni e al penultimo. Il primo bottone da sinistra consente di impostare un breakpoint nella posizione desiderata; l'elaborazione si fermerà in quel punto e consentirà all'utilizzatore di vedere il valore delle variabili, di eseguire porzioni di codice, ecc. In altre parole la *Command Window* diviene pienamente accessibile e consente ogni tipo di interazione con i dati, le funzioni e gli archivi su file.
Il secondo bottone cancella tutti i breakpoint.
Il terzo bottone ci fa avanzare di un'istruzione e fa bloccare nuovamente l'elaborazione per consentire una nuova indagine. Il penultimo bottone a destra forza l'elaborazione a proseguire sino al successivo breakpoint impostato oppure fino a fine programma.
Proviamo a seguire l'elaborazione di un programma: apriamo il programma MATLAB che calcola i coefficienti di correlazione (*edit corrcoef*) e piazziamo un breakpoint alla linea 77. Lanciamo ora il programma con un semplice comando sulla *Command Windows*:

```
>> corrcoef(rand(3,3))
```

A questo punto il programma si blocca e sulla *Command Window* compare un nuovo prompt *(K>>)*, indicante che ci troviamo in un altro modo di funzionamento dell'ambiente. Viene quindi attivato l'editor visualizzando il punto in cui il programma è stato fermato. Possiamo ora esplorare il contenuto del workspace per mezzo di:

```
K>> whos
  Name        Size          Bytes  Class

  varargin    0x0              0  cell array
  x           3x3             72  double array

Grand total is 9 elements using 72 bytes
```

Non dobbiamo aspettarci di vedere il workspace base di MATLAB (quello usato per richiamare la funzione *corrcoef*) ma invece quello locale della funzione. Ed in effetti troviamo il cell array vuoto dei parametri di input opzionali della funzione e la variabile *"x"* che è l'input richiesto dalla funzione.
A questo punto potremmo variare il valore di *"x"*, visualizzarlo o creare nuove variabili ausiliarie. Scegliamo invece di proseguire; la linea su cui siamo posizionati richiama una funzione presente nella porzione successiva del *m-file*. Per poterci entrare dobbiamo eseguire uno *step in*. Se posizioniamo il cursore del mouse sui bottoni che gestiscono il debug, e proviamo a scorrere bottone per bottone, vediamo apparire un piccolo riquadro giallo che riporta in chiaro l'azione che ogni singolo bottone esegue. Fermiamoci a quello che indica *Step in* e premiamolo.
Immediatamente ci vediamo proiettati in una nuova funzione che, per mezzo di quattro linee di codice, calcola i coefficienti di correlazione della matrice *"x"*. Se dopo alcune righe ci ritenessimo soddisfatti dell'esplorazione potremmo terminare la sessione di debug premendo il bottone che ha come descrizione *Continue* (riquadro giallo attivato dal passaggio del cursore del mouse). Questo comando forza MATLAB ad interrompere il debug e ad eseguire il codice sino al termine dell'elaborazione.
A questo punto il risultato del calcolo compare sulla *Command Window*.

IL PROFILER

Un'altra applicazione ausiliare molto utile è il *Profiler*. Essa è in grado di monitorare l'elaborazione di una catena di programmi per individuare dove venga speso tempo di elaborazione. Lo scopo è ovviamente quello di scoprire dove vi sia la possibilità di ridurre e ottimizzare la scrittura del codice con il fine di limare il più possibile i tempi di calcolo.
Per attivare il *Profiler* é necessario visitare il

☑ RIFERIMENTI

"Getting Started with MATLAB"
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

"Using MATLAB"
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/using_ml.pdf

"Using MATLAB Graphics"
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/graphg.pdf

menu *View* di MATLAB e quindi ciccare su *Profiler*. Si aprirà una finestra su cui spicca un bottone con la dicitura *Start Profiling*. Proviamo a premerlo e a lanciare sulla *Command Window* il comando:

```
>> corrcoef(rand(300,300));
```

Quindi premiamo nuovamente lo stesso bottone che ora riporterà la dicitura *Stop Profiling*. A questo punto apparirà una statistica circa i tempi esecutivi della catena di programmi che sono stati eseguiti per completare il compito che abbiamo imposto a MATLAB dalla *Command Window*.

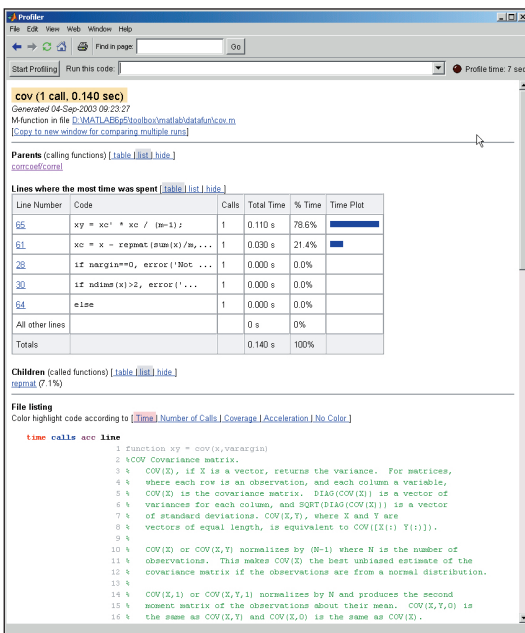



Fig. 8a: Statistiche del Profiler.

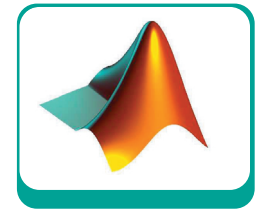
Proviamo a cliccare sul programma *cov* che pare essere uno di quelli che consumano più risorse (*corrcoef* è la funzione lanciata da *Command Window*, mentre *corrcoef/correl* è una sottofunzione di *corrcoef* che contiene proprio la funzione *cov*).
Quello che ci appare è riportato nelle Fig. 8a e 8b. Vediamo è una sintesi delle misure del tempo di elaborazione. Da qui individuiamo velocemente i punti più critici (Fig. 8a), mentre nel seguito viene riportato il codice con una vista analitica dei tempi (Fig. 8b), dalla quale scopriamo quali siano i punti sui quali possiamo intervenire per apportare miglioramenti significativi al codice con lo scopo di velocizzare il calcolo.
Esiste anche un modo procedurale di accedere alle funzioni di profiling. Se si prova a digitare *help profile* sulla *Command Vindow* si ottengono le notizie necessarie a comprendere come pilotare tale funzionalità.

```
Profiler
File Edit View Web Window Help
Find in page:                Go
Start Profiling  Run this code:            Profile time: 7 sec
                   21 %   See also CORRCOEF, STD, MEAN.
                   22
                   23 %   J. Little 5-5-86
                   24 %   Revised 6-9-88 LS 3-10-94 BJ
                   25 %   Copyright 1984-2002 The MathWorks, Inc.
                   26 %   $Revision: 5.16 $  $Date: 2002/06/05 17:06:38 $
                   27
0.000    1  .   28 if nargin==0, error('Not enough input arguments.'); end
0.000    1  .   29 if nargin>3, error('Too many input arguments.'); end
0.000    1  .   30 if ndims(x)>2, error('Inputs must be 2-D.'); end
                   31
0.000    1  .   32 nin = nargin;
                   33
                   34 % Check for cov(x,flag) or cov(x,y,flag)
         1  x   35 if (nin==3) | ((nin==2) & (length(varargin(end))==1));
                   36   flag = varargin(end);
                   37   nin = nin - 1;
0.000    1  .   38 else
0.000    1  .   39   flag = 0;
         1  .   40 end
                   41
         1  x   42 if nin == 2,
                   43   x = x(:);
                   44   y = varargin(1)(:);
                   45   if length(x) ~= length(y),
                   46     error('The lengths of x and y must match.');
                   47   end
                   48   x = [x y];
                   49 end
                   50
         1  x   51 if length(x)==prod(size(x))
                   52   x = x(:);
                   53 end
                   54
         1  x   55 [m,n] = size(x);
                   56
         1  x   57 if m==1,  % Handle special case
                   58   xy = 0;
                   59
         1  .   60 else
0.030    1  x   61   xc = x - repmat(sum(x)/m,m,1);  % Remove mean
         1  .   62   if flag
                   63     xy = xc' * xc / m;
0.000    1  .   64   else
0.110    1  .   65     xy = xc' * xc / (m-1);
         1  .   66   end
         1  .   67 end
                   68
```

Fig. 8b: Profiler.

CONCLUSIONI

In questo numero abbiamo esplorato molte delle applicazioni ausiliarie di MATLAB. Esse sono destinate a rendere il lavoro di coloro i quali devono sviluppare algoritmi il più semplice possibile. In molti casi si tratta di ausili al processo di sviluppo che sono volti ad aumentare in maniera considerevole la comodità di scrittura. In altri casi, come quello del *Profiler*, l'uso di questi strumenti migliora sensibilmente la qualità del prodotto finale consentendo l'ottimizzazione dei tempi di esecuzione.
Come sempre, invito tutti coloro i quali vogliano contribuire con suggerimenti e idee a contattarmi direttamente per mezzo dell'indirizzo di posta elettronica citato in calce. Per maggiori informazioni sui prodotti della famiglia MATLAB potete consultare il sito di The MathWorks (*www.mathworks.it*).
Suggerisco a tutti di guardare una porzione del sito web chiamato *"MATLAB Central"* (*www.mathworks.com/matlabcentral/*).
Esso riporta innumerevoli esempi d'uso di MATLAB in una varietà molto ampia di discipline tecnico scientifiche.

Fabrizio Sara
(fabrizio.sara@mathworks.it)

☑ PROFILER

Il Profiler puo' essere invocato dalla Command Window per mezzo di:

```
>> profile viewer
```

Applicazioni che interagiscono con effetti speciali

Un visualizzatore di immagini

In questo appuntamento completeremo l'applicazione Visualizzatore Immagini e, parallelamente, descriveremo come farla interagire con il "mondo esterno".

Nei precedenti articoli, dedicati alla gestione avanzata dei Form, abbiamo usato gli elementi dell'API di Windows per modificare il layout dei form e per creare degli effetti speciali durante la loro attivazione o disattivazione. Nei due precedenti appuntamenti, abbiamo avviato l'implementazione di un'applicazione, "laboratorio", con la quale abbiamo descritto come utilizzare i concetti che man mano esponevamo. L'applicazione è un "Visualizzatore Immagini", essa presenta un layout particolare e le sue "sottofinestre" compaiono e scompaiono con effetti speciali come esplosione ed implosione. Ricordiamo che abbiamo dato un aspetto particolare ai Form, utilizzando principalmente le funzioni dell'API della categoria *Region* (tra tutte citiamo la *CombineRgn*), mentre abbiamo impostato degli effetti speciali utilizzando le funzioni che consentono di manipolare oggetti *Rect* (cioè oggetti rettangolari) come per esempio la *GetWindowRect*; in particolare l'esplosione/implosione di un form è stata simulata disegnando aree rettangolari di estensione crescente/decrescente. I concetti esposti, naturalmente, possono essere usati per creare finestra di qualsiasi forma. In questo appuntamento completeremo il nostro Visualizzatore Immagini e parallelamente vedremo nuove caratteristiche del Visualizzatore Immagini di Windows XP; in particolare implementeremo le routine per la ricerca di immagini su internet e per gestire il ridimensionamento degli oggetti presenti sui form.

Innanzitutto, però, dobbiamo descrivere, sommariamente, le parti principali dell'applicazione, sia quelle già implementate che quelle da implementare.

Fig. 1: La finestra principale del Visualizzatore Immagini.

STATO DELL'ARTE

Il Visualizzatore Immagini è un'applicazione che permette di visualizzare file di immagini contenute in una directory. Le immagini possono anche essere scaricate da internet, come mostrere-

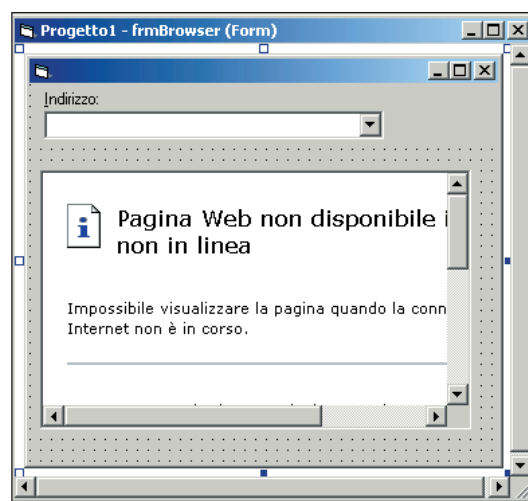



Fig. 2: Il form per connettersi ad Internet.

mo tra poco. L'interfaccia della finestra principale dell'applicazione è mostrata in Fig. 1. Essa presenta una barra di comando costituita di una serie di pulsanti che permettono di sfogliare le immagini, di proiettarle, di ricercarle sul computer o su internet, d'ingrandirle ecc. L'applicazione, per la ricerca di file nelle directory utilizza la finestra *FrmRicerca* (Fig. 2), mentre per la ricerca su Internet utilizza la finestra *FrmBrowser* (Fig. 3) o direttamente Internet Explorer, come mostreremo tra poco.
Ricordiamo che le immagini sul Form sono mostrate attraverso un *Frame* della libreria *MsForms*. Nel precedente articolo abbiamo descritto buona parte del codice dell'applicazione, lasciando in sospeso la ricerca di immagini su internet, il ridimensionamento della finestra principale e la gestione degli errori.
Iniziamo dalla ricerca delle immagini su internet, poi vedremo qualcosa sul ridimensionamento. Per quanto riguarda la gestione degli errori, daremo solo un cenno agli elementi di Visual Basic utilizzabili per lo scopo.

RICERCA SU INTERNET

Questa funzionalità può essere realizzata in due modi: utilizzando direttamente delle chiamate ad Internet Explorer oppure attraverso un form che contiene l'oggetto *WebBrowser*. Iniziamo a descrivere come utilizzare Internet Explorer.

```
Private Sub internet_Click()
 Dim IEApp As SHDocVw.InternetExplorer
 Set IEApp = New SHDocVw.InternetExplorer
 IEApp.Height = 500
 IEApp.Width = 500
 espandiform IEApp, 200
 IEApp.Visible = True
 IEApp.Navigate "http://www.edmaster.it"
End Sub
```

La procedura precedente permette di creare ed attivare un'istanza di Internet Explorer. Questo viene fatto attraverso gli elementi della libreria *SHDocVw (Internet Information Controls - SHDocVw.DLL)*. Se non volete referenziare la libreria potete utilizzare le seguenti istruzioni:

```
Dim IEapp As Object
Set IEapp = CreateObject("InternetExplorer.Application")
```

In questo secondo caso, per aprire ed attivare Internet Explorer, utilizziamo *CreateObject* con la stringa *"InternetExplorer.Application"*.

Nella procedura *internet_Click*, dopo aver creato un oggetto Internet Explorer ne fissiamo le dimensioni e lo facciamo apparire con l'effetto esplosione, utilizzando la procedura *espandiform* descritta nell'articolo precedente. Notate che in questo caso l'ID della finestra da passare alla *espandiform* è fornito da *IEApp* che non è un form, per questo l'interfaccia della *espandiform* deve essere modificata impostando il parametro *"frm"* di tipo *Object*, cioè:

```
Sub espandiform(frm As Object, passaggi As Long)
...
End Sub
```

LA FINESTRA CON IL WEBBROWSER

Per eseguire la ricerca delle immagini su internet, alternativamente, possiamo servirci dell'oggetto *WebBrowser* (che si trova nella libreria *SHDocVw.DLL*); per questo nel progetto introduciamo un nuovo form che chiamiamo *FrmBrowser*.
In esso inseriamo un oggetto *WebBrowser*, un *ComboBox* e una *PicturBox* come in Fig. 3 (in realtà bisognerebbe inserire anche la barra dei pulsanti, i menu a tendina, …). Con questo form, oltre ad illustrare un modo alternativo per proiettare la nostra applicazione su Internet vogliamo mostrare, come grazie alle *PictureBox* e ad una serie di accorgimenti, è possibile creare una finestra con oggetti "ridimensionabili". Nella Fig. 3 notate che il *Combo* (nominato indirizzo) si trova all'interno della *PictureBox* (nominata picindirizzo) e che quest'ultima è allineata al Top del form (valore della proprietà *Align=1 – Align Top*). Sul form gli elementi dovrebbero essere posti come in Fig. 3, altrimenti il codice, che tra poco definiremo, nell'evento *Resize*, non funziona correttamente.

Fig. 3: *FrmBrowser connesso ad internet.*

☑ L'OGGETTO ERR

Per gestire gli errori di Run-Time è importante conoscere le caratteristiche dell'oggetto Err, descriviamone brevemente le proprietà ed i metodi.

Number: restituisce o imposta il numero di errore;

Source: imposta o restituisce il nome dell'oggetto o dell'applicazione che ha generato l'errore;

Description contiene la descrizione dell'errore;

HelpFile restituisce o imposta un'espressione stringa che contiene il percorso completo relativo a un file della Guida del progetto;

HelpContext contiene un ID di contesto valido per il file della guida specificato.

I Metodi dell'oggetto Err sono:

Clear: azzera i valori di tutte le proprietà dell'oggetto Err;

Raise genera un errore di run-time.

Descriviamo le procedure a partire dalla *Form_Load*.

```
Private Sub Form_Load()
 On Error Resume Next
 Me.picIndirizzo.Align = 1
 Me.IntWebBrowser.Move picIndirizzo.Left, _
 picIndirizzo.Top + picIndirizzo.Height + 20
 indirizzo.Text = "http://www.edmaster.it"
 indirizzo_Click
End Sub
```

In questa procedura impostiamo il valore del *ComboBox* "indirizzo" e la posizione del WebBrowser e della *PictureBox*. Come accennato, con le istruzioni precedenti facciamo in modo che gli elementi del form si ridimensionino quando viene attivato l'evento *Resize* (attraverso la proprietà *WindowState*). Ricordiamo che il metodo *Move* serve a posizionare gli oggetti, esso ha la seguente sintassi:

```
oggetto.Move sinistra, superiore, larghezza, altezza
```

Naturalmente nel *Resize* dobbiamo prevedere del codice simile al seguente.

```
Private Sub Form_Resize()
 On Error Resume Next
 WBordo = (Me.Width - Me.ScaleWidth) / 2
 HTitolo = Me.Height - Me.ScaleHeight - WBordo
 indirizzo.Width = Me.Width - 2 * WBordo
 IntWebBrowser.Width = Me.Width - 2 * WBordo
 IntWebBrowser.Height = Me.Height - _
 picIndirizzo.Height - HTitolo - WBordo
End Sub
```

A livello globale dobbiamo definire le variabili:

```
Dim WBordo As Single, HTitolo As Single.
```

Ricordiamo che *ScaleWidth* fornisce il numero di pixel per l'asse orizzontale nel sistema di riferimento all'interno del Form (*ScaleHeight* per quello verticale) e che *Width* e *Heigth* per il form vengono calcolate comprendendo i bordi e la barra del titolo.
Oltre a queste procedure, naturalmente, prevediamo il codice per gestire la fase di inserimento dell'indirizzo del sito, da cui vogliamo scaricare le foto. Infatti, dopo essere state trovate su Internet, le foto devono essere scaricate in una directory (per esempio Immagini!) per essere, successivamente, sfogliate in modalità off-line.
L'indirizzo del sito, da consultare, lo connettiamo tramite le seguenti istruzioni:

```
Private Sub indirizzo_Click()
 IntWebBrowser.Navigate indirizzo.Text
End Sub
Private Sub indirizzo_KeyPress(KeyAscii As Integer)
 On Error Resume Next
 If KeyAscii = vbKeyReturn Then
  indirizzo_Click
 End If
End Sub
```

Oltre alle due procedure precedenti prevediamo il codice per mantenere, nel *ComboBox "indirizzo"*, gli URL già visitati. A tal fine definiamo la variabile *nonusare* che ci servirà per stabilire una condizione nella *indirizzo_Click*.
La *nonusare* verrà impostata dopo aver visitato un indirizzo, come sarà chiaro tra poco.

```
Dim nonusare As Boolean

Private Sub indirizzo_Click()
 If nonusare Then Exit Sub
 IntWebBrowser.Navigate indirizzo.Text
End Sub
```

Prima d'illustrare la procedura che carica gli URL nel *ComboBox indirizzi* descriviamo, l'evento del WebBrowser, *NavigateComplete2*. Esso si verifica dopo che il browser si è connesso con successo ad un nuovo indirizzo, quindi lo possiamo utilizzare per inserire gli URL nel *ComboBox* dopo che sono stati caricati. Di seguito presentiamo il codice per questo evento.

```
Private Sub intWebBrowser_NavigateComplete2(By
                         Val pDisp As Object, URL As Variant)
    On Error Resume Next
    Dim i As Integer
    Dim trovato As Boolean
    Me.Caption = IntWebBrowser.LocationName
    For i = 0 To indirizzo.ListCount - 1
        If indirizzo.List(i) = IntWebBrowser.LocationURL
                                                     Then
            trovato = True
            Exit For
        End If
    Next i
    nonusare = True
    If trovato Then
        indirizzo.RemoveItem i
    End If
    indirizzo.AddItem IntWebBrowser.LocationURL, 0
    indirizzo.ListIndex = 0
    nonusare = False
End Sub
```

Nella procedura precedente, oltre ad inserire gli indirizzi nel combo, impostiamo il valore della variabile *nonusare* che, come s'intuisce, serve per

☑ CATTURARE ERRORI

Per catturare gli errori in un'applicazione Visual Basic si possono utilizzare i seguenti elementi: On Error Goto <etichetta riga>, On Error Resume, On Error Resume Next, l'oggetto Err. Questi sono stati descritti ampiamente nei nostri precedenti articoli.

evitare di rivisitare un URL dopo aver impostato il valore da mostrare sul combo indirizzi (*indirizzo.ListIndex = 0*). Inoltre facciamo in modo che l'indirizzo che si sta visitando sia sempre nella prima posizione del combo; per questo sullo stesso elemento eseguiamo prima la *Remove* e poi la *Add*.
La procedura associata al pulsante Internet, della *FrmTrasparente*, che serve per attivare la *FrmBrowser* è la seguente.

```
Private Sub internet_Click()
 espandiform frmBrowser, 1000
 'descritta nel precdente articolo
 frmBrowser.Show 1
End Sub
```

Ora occupiamoci del ridimensionamento del form *FrmTrasparente*, cercando di non ingarbugliare le cose già definite nel precedente articolo.

FRMTRASPARENTE RESIZE

Dato che in generale un form trasparente è senza bordo conviene gestire soltanto il ridimensionamento innescato dalla proprietà WindowsState, che come è noto può assumere tre valori: 0=posizione normale, *1=miniminizzata* e *2=massimilizzata*. Il pulsante che attiva *WindowsState*, come si nota in Fig. 1, è posto in basso vicino al pulsante chiudi. La procedura associata al pulsante è la *pulsanteresize_Click*, nella quale oltre alla proprietà *WindowState* è gestita la proprietà *Picture* del pulsante (cioè l'immagine associata al pulsante).

```
Private Sub pulsanteresize_Click()
 If WindowState = 0 Then
  WindowState = 2
  pulsanteresize.Picture = LoadPicture(App.path +
                                   "/duerettangoli.bmp")
 Else
  WindowState = 0
  pulsanteresize.Picture = LoadPicture(App.path +
                                    "/unrettangolo.bmp")
 End If
End Sub
```

Le immagini *duerettangoli.bmp* e *unrettangolo .bmp* (questa ultima mostrata in Fig. 1) devono trovarsi nella directory del progetto.
Naturalmente questa funzione ha effetto, sugli elementi del form, se inseriamo del codice nell'evento *Resize*. Quindi nella *Form_Resize* dobbiamo prevedere del codice che posiziona gli elementi del form in base alle sue dimensioni (determinate da *WindowsState*). *Form_Resize* per esempio può essere la seguente.

```
Private Sub Form_Resize()
  Dim PuntoRife As Long
  WBordo = (Me.Width - Me.ScaleWidth) / 2
  HTitolo = Me.Height - Me.ScaleHeight - WBordo
  framefoto.Width = Me.Width - WBordo – 450
  framefoto.Height = Me.Height - Me.internet.Height -
  HTitolo - 450 - 50
  PuntoRife = framefoto.Height + 300
  bstop.Top = PuntoRife
  play.Top = PuntoRife
  avanti.Top = PuntoRife
  dietro.Top = PuntoRife
  zoomeno.Top = PuntoRife
  zoompiu.Top = PuntoRife
  internet.Top = PuntoRife
  cercafile.Top = PuntoRife
  pulsanteclose.Top = PuntoRife
  pulsanteresize.Top = PuntoRife
  disegnaform
  'disegnaform l'abbiamo descritta nel
  'precedente appuntamento
End Sub
```

Notate che 450 è la dimensione della barra di scorrimento (non inclusa in quella del framefoto) e che 50 serve per inserire dello spazio tra i bottoni e il *framefoto*.

Fig. 4: *Il Visualizzatore Immagini di Windows.*

Inoltre, per ridimensionare automaticamente anche l'immagine, presente nel framefoto, possiamo impostare la sua proprietà *PictureSizeMode* cioè:

```
framefoto.PictureSizeMode = fmPictureSizeModeZoom
```

☑ NAVIGATE

La sintassi del metodo navigate è la seguente IE.Navigate URL [Flags,] [TargetFrameName,] [PostData,] [Headers] I parametri principali sono: URL che rappresenta il documento da caricare. Flags che specifica se aggiungere l'URL visitato alla history list. TargetFrameName che è il frame in cui la pagina verrà mostrata.

Questa proprietà ammette tre valori: *fmPictureSizeModeStretch*, *fmPictureSizeModeZoom* e *fmPictureSizeModeClip*. La *PictureSizeMode* di default è impostata su *fmPictureSizeModeClip*.

LO SCROLLING MANUALE

Per rendere più semplice lo spostamento delle immagini sul framefoto, oltre alle scrollbar possiamo utilizzare il metodo *Scroll* ed attivarlo attraverso una combinazione di tasti come per esempio i tasti freccia.
Innanzitutto illustriamo la sintassi del metodo *Scroll*:

```
object.Scroll( [ ActionX [, ActionY]])
```

I due parametri opzionali, *ActionX* e *ActionY*, rappresentano rispettivamente l'azione che verrà fatta sull'asse orizzontale e su quello verticale.
I valori che questi parametri possono assumere sono simili a quelli dei parametri nell'evento *FrameFoto_Scroll*, descritto nell'articolo precedente, ed in parte riportati nella seguente:

Stringa	Valore	Descrizione
fmScrollActionNoChange	0	Nessuna azione
fmScrollActionLineUp	1	Spostamento a sinistra o verso l'alto
fmScrollActionLineDown	2	Spostamento a destra o verso il basso

Tab. 1: Alcuni valori delle proprietà Action.

Il codice per gestire il movimento lo inseriamo nella procedura *PicMove* che richiamiamo dall'evento *framefoto_KeyDown*.

```
Private Sub framefoto_KeyDown(KeyCode
               As MSForms.ReturnInteger, Shift As Integer)
 PicMove (KeyCode)
End Sub

Sub PicMove(KeyCode As Integer)
 Select Case KeyCode
  Case 37
   'freccia a sinistra
   framefoto.Scroll 1, 0
  Case 38
   'freccia sopra
   framefoto.Scroll 0, 1
  Case 39
   'freccia a destra
   framefoto.Scroll 2, 0
  Case 40
   'freccia sotto
   framefoto.Scroll 0, 2
 End Select
End Sub
```

☑ RUNDLL32

Per avviare il visualizzatore immagini di Windws XP dal Prompt dei comandi, oppure tramite il programma Esegui, possiamo usare una riga di comando come la seguente:

RunDll32.exe shimgvw.dll, ImageView_Fullscreen c:\prova.bmp

Questo comando apre il Preview di Windows XP a pieno schermo e mostra l'immagine "prova.bmp"

IL VISUALIZZATORE IMMAGINI DI XP

Concludiamo con tre istruzioni a dir poco sconvolgenti!?

```
Dim comodo As String
comodo = "RunDll32.exe shimgvw.dll,
                     ImageView_Fullscreen " + "c:\foto"
Shell comodo, vbNormalFocus
```

Se inserite queste istruzioni in un *CommandButton* potrete consultare i file di immagini (di svariati tipi) contenuti nella cartella *"c:/foto"*. Vi consigliamo di studiare attentamente queste istruzioni e di capire come poterle usare nelle vostre applicazioni.
Ricordiamo che *Shell* avvia un eseguibile e restituisce il suo ID.
La sintassi di *Shell* è la seguente

```
Shell(pathname[,windowstyle])
```

Pathname è il nome del programma da eseguire, *windowstyle* serve per specificare lo stile della finestra che mostra il programma.
I valori della *windowstyle* sono riportati nella tabella seguente:

Costante	Valore	Descrizione
vbHide	0	La finestra è attiva e nascosta
vbNormalFocus	1	La finestra è attivata nella posizione originale.
vbMinimizedFocus	2	La finestra è attiva ad icona.
vbMaximizedFocus	3	La finestra è attiva e ingrandita.
vbNormalNoFocus	4	La finestra è nella posizione originale ma non attiva.
vbMinimizedNoFocus	6	La finestra è ridotta ad icona ma non è attiva.

Tab. 2: I valori di windowstyle.

CONCLUSIONI

In questo appuntamento abbiamo chiarito diversi aspetti sulla gestione dei form e parallelamente abbiamo aggiunto nuove caratteristiche all'applicazione *Visualizzatore Immagini.*
Nei successivi appuntamenti ci occuperemo di argomenti caldi come *HotKey*, gestione porte parallele ed USB … Seguiteci!

Massimo Autiero

Web Service: funzionalità avanzate

Attachment in SOAP con Delphi 7

SOAP ha rappresentato una vera e propria rivoluzione nel mondo dell'interoperabilità di servizi applicativi esposti da sistemi eterogenei. Quello che mancava era il supporto agli attachment; la specifica è stata finalmente completata e Borland, con la versione 7 di Delphi l'ha implementata.

A mio avviso SOAP è davvero una delle più grandi rivoluzioni informatiche degli ultimi anni. Probabilmemte non è ancora recepita ed utilizzata fino in findo per quello che è, ma spesso viene vista come una delle tante tipologie di applicazioni che si possono creare da Visual Studio .NET o da Delphi, ma non come il protocollo standard di fruizione di oggetti da sistemi eterogenei. Sarà quindi un processo lento di assimilazione e di presa di coscienza, ma sicuramente inesorabile. Quello che finora mancava era il supporto agli allegati, cioè a file e stream che si possono inviare al server o ricevere da questo. In effetti non è facile inviare in formato xml una cartina raster di una planimetria o il database Access dei clienti, ma non è nemmeno impossibile considerando che da anni, senza dare troppo importanza alla cosa, facciamo dal browser il download o l'upload via http di file. Il principio è semplice: il file binario viene trasferito come se fosse un normale file di testo, cioè ogni byte che lo compone viene trattato come un carattere stampabile e poi viene riassemblato a destinazione. Il vero problema dei binari è come rappresentare in xml o in testo libero i caratteri non stampabili, quelli che praticamente rendono problematica, ad esempio, la lettura di file binari aperti con Notepad. Fortunatamente esiste un formato di trasformazione che si chiama *base64* che si basa su un principio molto semplice: per ogni byte che compone il file binario, se questo corrisponde ad un carattere stampabile, allora lo si lascia inalterato, se invece non è stampabile, si trasforma in una sequenza di *n* caratteri stampabili corrispondenti a quel byte. Stesso discorso vale per eventuali byte corrispondenti a caratteri stampabili incompatibili con xml o html: in questo caso vengono nuovamente sostituiti con una sequenza di caratteri stampabili innocui. L'effetto positivo è di ottenere un unico grande testo, quello negativo è che la dimensione in byte del testo può essere anche molto maggiore (persino del 50%) della dimensione in byte del binario originario. Ovviamente lo stream base64 viene riconvertito con un processo contrario quando arriva a destinazione e il file binario viene così ricostruito.

LA SPECIFICA SOAP 1.1

La specifica SOAP 1.1, la prima rilasciata ad avere una reale implementazione, non conteneva nessun riferimento alla possibilità di inviare binari. Solo successivamente è stata messa a punto la specifica SOAP con attachment che prevede essenzialmente di utilizzare il protocollo MIME, lo stesso utilizzato nella posta elettronica. In pratica si tratta di una sovrastruttura sul protocollo base64. *MIME multipart,* però, prevede alcune scelte progettuali che rendono la sua implementazione poco efficiente, almeno a detta di alcuni. Questo resterebbe soltanto un giudizio di parte e persino sterile, se non fosse che è condiviso, anzi propugnato da uno dei massimi ideatori ed implementatori della specifica SOAP: Microsoft. Infatti, nonostante siano stati tra gli autori della specifica, al momento della reale implementazione in .NET e nella versione 3 del Soap Toolkit per COM hanno deciso di avallare una nuova specifica standard di cui loro stessi, ancora una volta, sono stati tra gli ideatori cioè DIME *(Direct Internet Message Encapsulation).* Anch'essa usa base64 per l'encoding/decoding degli allegati, ma introduce un protocollo completamente diverso per tutta la parte di header. Gli autori sostengono che sia più semplice, più potente e che renda meno probabili

errori di implementazioni che ne inficino l'interoperabilità. Intanto, al di la di quella che sia la verità, il problema è che ci sono due differenti specifiche e quel che è peggio è che le suite SOAP attualmente disponibili abbiano deciso di implementare l'una piuttosto che l'altra. Microsoft .NET implementa DIME, Borland BizSnap implementa MIME multipart… Quindi? La brutta notizia è che non sono interoperabili. Suppongo, ma sono quasi certo, che saranno Borland e tutti gli altri implementatori pro MIME a piegarsi in futuro, coprendo anche DIME perché attualmente non si può pensare di implementare SOAP rendendosi incompatibili con Microsoft e la sua corazzata .NET.

BORLAND BIZSNAP DI DELPHI 7

Borland ha introdotto con Delphi 6 la tecnologia BizSnap che altro non è che l'implementazione della specifica SOAP 1.1. All'epoca dell'uscita di Delphi 6 nemmeno .NET era stato ancora rilasciato, era in beta, in giro si trovavano alcuni toolkit SOAP più o meno efficaci e Sun, IBM e tutti quelli che "bazzicano" Java erano ancora a rigirarsi i pollici mentre riflettevano se e come supportare Java. Borland, invece, era già pronta e il suo prodotto era fantastico: completo, semplice e potente.

Nome	Tipo	Descrizione
ContentType	Proprietà	Legge e imposta il ContentType MIME dell'allegato
Encoding	Proprietà	Legge e imposta l'alfabeto di encoding dell'allegato. Il default è lo standard UTF-8
Headers	Proprietà	Ritorna o imposta il TStrings contentente gli header MIME dell'allegato.
Ownership	Proprietà	Ritorna od imposta il tipo di controllo sullo stream (TStream) che contiene l'allegato vero e proprio. Si tratta di un tipo enumerato che può contenere i valori soReference (lo stream è solo riferito e non viene distrutto alla distruzione del TSOAPAttachment) o soOwned (lo stream ha scope solo nell'ambito del TSOAPAttachment).
SourceStream	Proprietà	Il TStream interno.
SourceString	Proprietà	L'encoding in base64, secondo l'alfabeto descritto da Encoding, quindi rappresenta proprio la stringa di encoding dell'allegato.
ObjectToSOAP	Metodo	Converte un allegato nella sua rappresentazione SOAP.
SaveToFile	Metodo	Salva l'allegato SOAP su un file
SaveToStream	Metodo	Salva l'allegato SOAP in uno stream
SetSourceFile	Metodo	Imposta il file di origine dell'allegato SOAP, in pratica il file che si intende spedire
SetSourceStream	Metodo	Imposta lo stream di origine dell'allegato SOAP, quindi lo stream da spedire
SOAPToObject	Metodo	Converte un allegato SOAP in un oggetto.

Tab. 1: L'oggetto TSOAPAttachment.

Certo la specifica non era ancora consolidata e quindi l'interoperabilità era ancora un sogno nemmeno troppo confessabile. Tant'è che i tre service pack che si sono succeduti su Delphi 6 erano in realtà dei service pack per BizSnap, non tanto per correggerne i bug, ma per renderlo pienamente interoperabile. L'obiettivo è stato raggiunto ed infatti BizSnap Update Pack 3 interagisce con pochissimi o nessun problema con .NET, IBM WebSphere e quasi tutti gli altri, come ho mostrato in alcuni miei precedenti articoli. Cosa ci si poteva aspettare da Delphi 7, allora? Un'interoperabilità ancora più completa e senza l'uso di alcuni trucchetti come avveniva per la 6 e soprattutto l'implementazione di ciò che di SOAP mancava ancora in *BizSnap: i SOAP Header e i SOAP Attachment.* Questi secondi, come avrete potuto ampiamente già "sospettare", ci occuperemo in questo articolo. Per una migliore comprensione dell'articolo sarebbe necessaria una conoscenza, seppur minima di BizSnap o almeno dei concetti legati a SOAP. La rete è ricchissima di materiale, ma anche le edicole e le librerie… Sia per i neofiti che per coloro che già conoscono SOAP e/o BizSnap è interessante notare come è strutturato un messaggio SOAP contentente un attachment MIME multipart; si immagini di voler invocare un servizio strutturato col seguente prototipo in linguaggio Delphi:

```
function GetFile(username, password, filename :
                              String) : TSOAPAttachment;
```

Si tratta di un semplice metodo che richiede al server un certo allegato. Infatti, il valore di ritorno del metodo è un oggetto *TSOAPAttachment* che è la rappresentazione BizSnap di un allegato MIME multipart. In Tab. 1 è possibile osservare la firma dell'oggetto *TSOAPAttachment*. L'oggetto, come tutti gli oggetti che sono rappresentazioni di parametri o oggetti da trasmettere o ricevere via SOAP, discende da *TRemotable*. Internamente rimappa un *TStream* che, concettualmente, è la cosa più vicina ad un attachment SOAP, cioè uno stream binario di dati eterogenei.
Questo ci consente di creare un Soap attachment a partire da uno stream e di deserializzare un attachment in uno stream.
Osserviamo la richiesta SOAP:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
 "http://schemas.xmlsoap.org/soap/envelope/"xmlns:xsd=
    "http://www.w3.org/2001/XMLSchema" xmlns:xsi=
       "http://www.w3.org/2001/XMLSchema-instance"
                                  xmlns:SOAP-ENC=
          "http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body SOAP-ENV:encodingStyle=
          "http://schemas.xmlsoap.org/soap/encoding/">
    <NS1:GetFile xmlns:NS1="urn:
      AttachmentsManagerIntf-IAttachmentsManager">
      <username xsi:type="xsd:string">ciro</username>
      <password xsi:type="xsd:string">ciro</password>
      <filename xsi:type="xsd:string">
                              MSSCCPRJ.SCC</filename>
    </NS1:GetFile>
```

```
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fin qui, per chi ha già osservato altre volta come è strutturata una richiesta SOAP, non c'è nessuna novità. Le novità si incontrano ovviamente nella risposta del server che dovrà contenere anche l'allegato MIME multipart richiesto:

```
Content-ID: <http://www.borland.com/rootpart.xml>
Content-Location: http://www.borland.com/rootpart.xml
Content-Length: 538
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd=
    "http://www.w3.org/2001/XMLSchema" xmlns:xsi=
      "http://www.w3.org/2001/XMLSchema-instance"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/
                              soap/encoding/">
  <SOAP-ENV:Body SOAP-ENC:encodingStyle=
      "http://schemas.xmlsoap.org/soap/envelope/">
    <NS1:GetFileResponse xmlns:NS1="urn:
      AttachmentsManagerIntf-IAttachmentsManager">
      <return href="cid:EF58639D-EB0E-4F7E-
                              A92B-D23397264C49"/>
    </NS1:GetFileResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
—MIME_boundaryB0R9532143182121
Content-ID: <EF58639D-EB0E-4F7E-A92B-D23397264C49>
Content-Length: 346
Content-Type: application/binary
[SCC]
SCC=This is a source code control file
[Scripting.vbp]
SCC_Project_Name=this project is not under source
                                     code control
SCC_Aux_Path=<This is an empty string for the
                                  mssccprj.scc file>
[PassingObjects.vbp]
SCC_Project_Name=this project is not under source
                                     code control
SCC_Aux_Path=<This is an empty string for the
                                  mssccprj.scc file>
—MIME_boundaryB0R9532143182121—
```

Innanzitutto la risposta è preceduta da un tipico header http che preannuncia alcune informazioni relative alla risposta contenente l'allegato. Subito dopo, però, riconoscerete sicuramente il tradizionale *envolope SOAP Response*. Questo ha particolarità: il tag *<return>* fa riferimento ad un file allegato raggiungibile http, cioè *href="cid:EF58639D-EB0E-4F7E-A92B-D23397264C49".* Infatti la parte successiva del messaggio è proprio un file MIME multipart con lo stesso *Id*. Il suo Content /Type è il tradizionale (ed indeterminato) application/binary e la sua lunghezza è 346 byte (caratteri). A questo punto

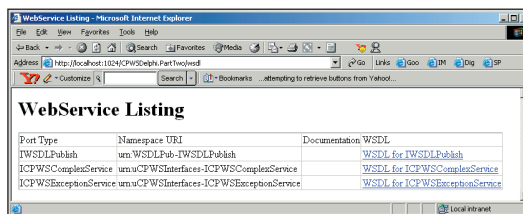



Fig. 1: ***Il Web App Debugger visto dal browser.***

troviamo finalmente il contenuto del nostro allegato che, nell'esempio, è un file in chiaro. Se si fosse trattato di un file binario puro avreste trovato una incomprensibile sequenza di caratteri base64. L'allegato inizia e termina con il tag *—MIME_boundaryB0R9532143182121*. Semplice, ma efficace, da far venir voglia di implementarsi MIME multipart da zero e senza l'uso di alcun toolkit. Ma questa idea balzana vi passa quasi subito se osservata quanto è banale utilizzare tutta questa roba in Delphi 7. Ma andiamo con ordine.

UN ESEMPIO REALE

Siccome comprendere la teoria è importante, ma vederne un'applicazione reale è spesso più efficace, immaginiamo di voler realizzare un web service che faccia da repository remoto di nostri file, una sorta di ftp basato su SOAP. Certo, poteva trovare un'idea più intelligente, ma l'obiettivo e mostrare la tecnologia e le sue potenzialità: le implementazioni reali e plausibili sono lasciate al lettore… Implementeremo il nostro web service come un'applicazione di tipo *Web App Debugger* in modo da non rendere necessaria la presenza di un web server sulla nostra macchina durante le fasi di scrittura del codice e per poter facilmente effettuare il debug del codice durante la scrittura, cosa possibile solo attraverso la tecnologia *Web App Debugger* (Box laterale). Per le fasi di creazione di un web service con Delphi 6/7 e per la scelta della tipologia di applicazione SOAP da creare si rimanda alla documentazione specifica.
Il nostro file server deve essere dotato dei seguenti metodi raccolti nella tipica interfaccia che deve essere sempre definita quando si intende produrre un web service con Delphi BizSnap:

```
IAttachmentsManager = interface(IInvokable)
['{1FCAC702-CD3C-4260-BF05-743EECEB3C3C}']
  function GetFileList(username, password : String)
                              : TStringArray; stdcall;
  function GetFile(username, password, filename
                  : String) : TSOAPAttachment; stdcall;
  function UploadFile(username, password, filename: String;
         fileToUpload : TSOAPAttachment) : Boolean; stdcall;
end;
```

Il metodo *GetFile*, che abbiamo già introdotto in

☑ DEBUGGING DI APPLICAZIONI WEB SENZA WEB SERVER

Borland ha deciso di fornire anche la possibilità di testare le applicazioni Web (WebSnap e BizSnap) anche in mancanza di un web server. Per far ciò ha introdotto un tool con Delphi 6 che si chiama Web App Debugger, esso permette di mandare in esecuzione, e di effettuare il debugging, di queste applicazioni anche senza il server http. L'idea è ingegnosa: si realizza una nuova applicazione web, ma questa volta di tipo Web App Debugger executable. In questo modo viene realizzato un oggetto COM ActiveX Exe (si, avete letto bene) che implementa un'interfaccia convenzionale. Al progetto verrà infatti aggiunta automaticamente una unit con un form nella cui parte di inizializzazione viene istanziato l'oggetto COM, cioè l'applicazione stessa:

```
implementation
uses ComApp;
const
  CLASS_ComWebApp:
    TGUID = '{B1076694-
        49C4-4CC1-BD70-
         66F83908F851}';
initialization
  TWebAppAutoObject
          Factory.Create(
       Class_ComWebApp,
    'PartTwo', 'PartTwo
               Object');
```

Fig. 2: *La console del Web App Debugger.*

☑ DEBUGGING DI APPLICAZIONI WEB SENZA WEB SERVER

Il *Web App Debugger* può essere mandato in esecuzione (Fig. 2), mettendosi in ascolto sulla porta 1024 locale (o su altre configurabili ad hoc). Questo tool, tra le altre funzionalità, permette anche di tracciare un log completo di tutte le chiamate http al server. Infatti, accedendo all'URL:

http://localhost:1024/ServerInfo.ServerInfo

si richiamerà un eseguibile Web App Debugger di sistema, con *ProgID ServerInfo. ServerInfo*, che ci mostrerà la mappa delle applicazioni WebApp installate nel sistema locale, tra cui la nostra (con *ProgID CPWSDelphi.PartTwo*). Così possiamo finalmente far eseguire la nostra applicazione da *WebAppDebugger* e farci restituire il WSDL invocando la URL:

http://localhost:1024/CPWSDelphi.PartTwo/wsdl

E' inutile dire che poter effettuare il debugging step by step, è impareggiabile. Basta mettere in Run l'applicazione con un F9 e il gioco è fatto. Finalmente le stringhe di debugging stampate direttamente nella pagina web del client saranno solo un ricordo (triste)…

precedenza per spiegare la messaggistica SOAP, si occupa di tirare giù un allegato dal server identificato da un nome. I parametri *username* e *password* consentono di effettuare questa operazione solo se si è autorizzati. In realtà la gestione degli account nell'esempio è davvero primitiva. Il metodo *GetFileList* restituisce invece un array di stringhe contenente l'elenco di tutti i file presenti sul server. Infine *UploadFile* consente di fare l'upload di un nuovo file sul server che, da quel momento, diventerà disponibile per un successivo download. Il web service è definito nel progetto *SoapAttachmentes.dpr* disponibile tra i sorgenti allegati al presente articolo. Per comprendere l'assoluta semplicità di gestione degli allegato SOAP con la libreria *BizSnap*, osserviamo l'implementazione dei metodi dell'interfaccia nella classe *TAttachmentsManager* del progetto a cominciare dalla *GetFile*:

```
const FILE_PATH : String = 'c:\prove\';
const STD_USERNAME : String = 'ciro';
const STD_PASSWORD : String = 'ciro';
function TAttachmentsManager.GetFile(username,
        password, filename: String): TSOAPAttachment;
var
  stream : TFileStream;
begin
  if (username = STD_USERNAME) and (password =
                                  STD_PASSWORD) then
  begin
    stream:= TFileStream.Create(FILE_PATH +
                                 filename, fmOpenRead);
    Result := TSoapAttachment.Create;
    Result.SetSourceStream(stream, soOwned);
  end
  else
    Raise Exception.Create('Login errata!');
end;
```

Dopo aver verificato la login, confrontandola banalmente con due costanti definite nel codice, viene caricato il file richiesto in un *TFileStream* (un discendente di *TStream* che lavora sul filesystem). Viene dunque creato il *TSOAPAttachment* che costituirà il valore di ritorno della funzione, nonché l'allegato da inviare al client e lo stream verrà assegnato al *TSOAPAttachment* con l'istruzione *SetSourceStream*. E il gioco è fatto…

Il metodo *UploadFile*, invece, funziona al contrario: consente ad un client di inviare un allegato sul server. Osserviamone l'implementazione:

```
function TAttachmentsManager.UploadFile(username,
                         password, filename: String;
  fileToUpload: TSOAPAttachment): Boolean;
begin
  Result:= false;
  if (username = STD_USERNAME) and (password =
                                  STD_PASSWORD) then
  begin
    fileToUpload.SaveToFile(FILE_PATH + filename);
    Result:= true;
  end
  else
    Raise Exception.Create('Login errata!');
end;
```

L'implementazione di questo metodo è ancora più banale: il *TSOAPAttachment*, giunto come parametro dal client, viene salvato direttamente nella cartella del server in cui vengono conservati i file esposti ai client. Ma, come abbiamo già fatto per *GetFile*, osserviamo come, a basso livello, viene assemblata ed inviata la richiesta *UploadFile* dal client al server:

```
Content-Type: text/xml
SOAPAction: "urn:AttachmentsManager
                    Intf-IAttachmentsManager#UploadFile"
Content-ID: <http://www.borland.com/rootpart.xml>
Content-Location: http://www.borland.com/rootpart.xml
Content-Length: 700
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd=
    "http://www.w3.org/2001/XMLSchema" xmlns:xsi=
   "http://www.w3.org/2001/XMLSchema-instance" xmlns:
 SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body SOAP-ENV:encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/">
  <NS1:UploadFile xmlns:NS1="urn:
     AttachmentsManagerIntf-IAttachmentsManager">
    <username xsi:type="xsd:string">ciro</username>
    <password xsi:type="xsd:string">ciro</password>
    <filename xsi:type="xsd:string">Listener.asp
                                          </filename>
    <fileToUpload href="cid:2477FD62-6EB3-4655-B6F2-
                                    D08AC06A26DA"/>
  </NS1:UploadFile>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
—MIME_boundaryB0R9532143182121
Content-ID: <2477FD62-6EB3-4655-B6F2-D08AC06A26DA>
Content-Length: 141
Content-Type: application/binary
Content-transfer-encoding: binary
<%
Dim oXMLProxy
```

```
Set oXMLProxy = CreateObject(
                    "AmarcordApplicationServer.Processor")
oXMLProxy.ProcessRequest Request, Response
%>
—MIME_boundaryB0R9532143182121—
```

Si può osservare come, ancora una volta, il parametro *fileToUpload* di tipo *TSOAPAttachment*, cioè proprio il file che il client invia al server, sia a livello SOAP definito semplicemente come un link ed in particolare come il link HTTP *href="cid: 2477FD62-6EB3-4655-B6F2-D08AC06A26DA"*.
Al termine dell'evenlope SOAP viene riportato proprio il file allegato in formato MIME multipart, analogamente a quanto descritto in precedenza.
La risposta del server sarà un più banale:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
 "http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd=
      "http://www.w3.org/2001/XMLSchema" xmlns:xsi=
    "http://www.w3.org/2001/XMLSchema-instance" xmlns:
 SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body SOAP-ENC:encodingStyle=
           "http://schemas.xmlsoap.org/soap/envelope/">
  <NS1:UploadFileResponse xmlns:NS1="urn:
       AttachmentsManagerIntf-IAttachmentsManager">
    <return xsi:type="xsd:boolean">true</return>
  </NS1:UploadFileResponse></SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In pratica un *true* o un *false* che indica la riuscita o meno dell'upload, come si poteva evincere dall'implementazione Delphi del metodo *UploadFile*. Il metodo *GetFiles*, invece, non presenta particolarità interessanti ai fini dell'obiettivo di questo articolo, per cui per la sua consulatazione si rimanda al sorgente allegato.

IL CLIENT DEL NOSTRO WEB FILE SERVER

Il progetto *SoapAttachmentsClient.dpr* rappresenta il client che consuma il nostro servizio. Per l'aggancio e l'importazione di un web service da un client Delphi si rimanda alla documentazione specifica. Osserviamo come si presenta in Fig. 3.
Si tratta di una semplice applicazione Windows scritta in Delphi che si connette al nostro web service e richiede immediatamente l'elenco dei file presenti sul server attraverso il metodo *GetFiles* del web service:

```
var
  attachManager : IAttachmentsManager;
```

```
  files : TStringArray;
  i : Integer;
begin
  Memo1.Clear;
  Memo2.Clear;
  attachManager:= GetIAttachmentsManager(
                          false, cbURL.Text, HTTPRIO1);
  files:= attachManager.GetFileList(txtUsername.Text,
                                        txtPassword.Text);
  ListBox1.Clear;
  for i:= 0 to Length(files) - 1 do
  begin
    ListBox1.Items.Add(files[i]);
  end;
```

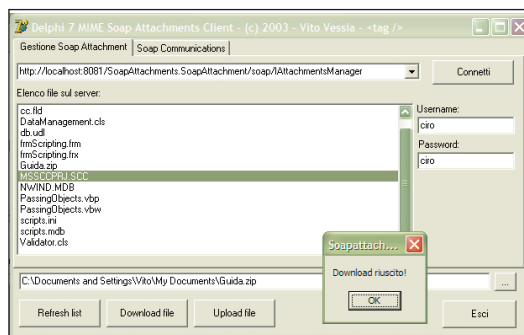



Fig. 3: Il client in azione.

Da notare come il web service, o meglio il suo proxy lato client, venga creato dalla funzione *GetIAttachmentManager* che è stata generata automaticamente da wizard di importazione lato client di Delphi per i web service. L'elenco viene mostrato in una listbox e l'utente, semplicemente con un doppio click sull'elemento desiderato, può richiedere il download di quel file (attraverso l'invocazione del metodo *GetFile* sull'evento *DoubleClick* della listbox):

```
procedure TForm1.ListBox1DblClick(Sender: TObject);
var
  attachManager : IAttachmentsManager;
  soapAttach : TSOAPAttachment;
begin
  if ListBox1.ItemIndex <> - 1 then
  begin
    attachManager:= GetIAttachmentsManager(
                          false, cbURL.Text, HTTPRIO1);
    soapAttach:= attachManager.GetFile(txtUsername.Text,
     txtPassword.Text, ListBox1.Items[ListBox1.itemindex]);
    CreateDir(ExtractFileDir(Application.ExeName)
                                          + '\downloads');
    soapAttach.SaveToFile(ExtractFileDir(
           Application.ExeName) + '\downloads\' +
             ListBox1.Items[ ListBox1.itemindex]);
    ShowMessage('Download riuscito!');
  end;
end;
```

☑ RIFERIMENTI

RFC ufficiale della specifica base64
http://www.ietf.org/rfc/rfc2045.txt

Documento w3c ufficiale della specifica SOAP con attachment
http://www.w3.org/TR/SOAP-attachments

Documento della specifica DIME
http://gotdotnet.com/team/xml_wsspecs/dime/

http://community.borland.com/article/0,1410,27301,00.html

Anche l'upload è altrettanto semplice: si sceglie da una finestra *Common Dialog* il file da inviare e lo si invia.
Eccone il codice:

```
procedure TForm1.cmdUploadFileClick(Sender:TObject);
var
  attachManager : IAttachmentsManager;
  soapAttach : TSOAPAttachment;
begin
  if txtUploadFile.Text <> '' then
  begin
     attachManager:= GetIAttachmentsManager(false,
                                   cbURL.Text, HTTPRIO1);
     soapAttach:= TSOAPAttachment.Create;
     soapAttach.SetSourceFile(txtUploadFile.Text);
     attachManager.UploadFile(txtUsername.Text,
                    txtPassword.Text, ExtractFileName(
                      txtUploadFile.Text),  soapAttach);
  end
  else
     ShowMessage('File non definito!');
end;
```

Una interessante possibilità offerta da *BizSnap* di Delphi 7 è l'intercettazione della messaggistica SOAP che viene inviata e ricevuta dal server.
In pratica l'oggetto *THTTPRIO*, che funge da proxy verso il web service, adesso espone quattro eventi che si scatenano quando viene inviata o ricevuta una richiesta SOAP o quando viene inviato o ricevuto un SOAP Attachment. Per l'esempio sono stati intercettati i primi due eventi di cui possiamo osservare i prototipi:

```
procedure HTTPRIO1AfterExecute(const MethodName:
                          String; SOAPResponse: TStream);
procedure HTTPRIO1BeforeExecute(const
  MethodName: String; var SOAPRequest: WideString);
```

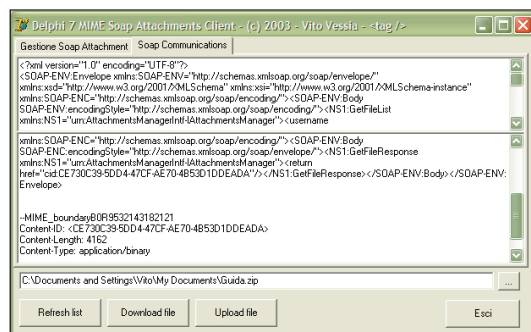



Fig. 4: ***La messaggistica SOAP "spiata" dal client.***

Con il primo vengono intercettati i messaggi SOAP che il client invia al server, con il secondo invece le risposte del server. Grazie a questo due eventi è possibile spiare SOAP a livello più basso per capire come funziona davvero. Il contenuto ricevuto dai due eventi viene rediretto su due *TMemo* presenti in un altro pannello del form del client di esempio, come mostrato in Fig. 4.
Osserviamone la semplice implementazione degli event handler:

```
procedure TForm1.HTTPRIO1AfterExecute(const
        MethodName: String; SOAPResponse: TStream);
var
  Temp: TStringStream;
begin
  Temp := TStringStream.Create('');
  try
    Temp.CopyFrom(SOAPResponse, 0);
    Memo2.Text := Temp.DataString;
  finally
    Temp.Free;
  end;
end;
procedure TForm1.HTTPRIO1BeforeExecute(const
  MethodName: String; var SOAPRequest: WideString);
begin
  Memo1.Lines.Add(SOAPRequest);
end;
```

CONCLUSIONI

Questo articolo ha introdotto e spiegato alcuni concetti legati alla gestione degli allegati della specifica SOAP, almeno nella versione più antica ed ufficiale, MIME multipart, che è stata scelta da Borland per la sua implementazione nella libreria SOAP BizSnap presente in Delphi 7. Inoltre abbiamo creato un esempio reale osservando la semplicità di fruizione e di scrittura del codice per realizzare una funzionalità che, tutto sommato, non si può considerare banale. E abbiamo cercato di spiare cosa accadeva a basso livello cioè come quelle semplici istruzioni Delphi venissero trasformate in messaggi SOAP che, non dimentichiamoci, è un protocollo standard di fruizione di servizi web da piattaforme eterogene basato sul protocollo HTTP per la comunicazione e su un xml governato da schemi standard per la parte applicativa.
Ciò significa che avremmo potuto scrivere un client che fruiva del servizio di file server anche in una piattaforma diversa da Delphi, così come in Delphi avremmo potuto scrivere un client in grado di consumare un servizio SOAP che, tra i suoi parametri, permetta di inviare o ricevere allegati, purchè di tipo MIME multipart. In soldoni: il server sarebbe potuto essere un host AS/400 con Web Sphere e, se non ci fosse stata di mezzo la diatriba tra MIME multipart e DIME, anche un server Windows con IIS e un web service .NET.
Sembra un sogno che si trasforma in realtà…

Vito Vessia

☑ L'AUTORE

Vito Vessia è cofondatore della codeBehind S.r.l.

http://www.codeBehind.it

una software factory di applicazioni enterprise, web e mobile, dove progetta e sviluppa applicazioni e framework in .NET, COM(+) e Delphi occupandosi degli aspetti architetturali. È autore del libro "Programmare il cellulare", Hoepli, 2002, sulla programmazione dei telefoni cellulari connessi al PC con protocollo standard AT+. Può essere contattato tramite e-mail all'indirizzo

vito.vessia@ioprogrammo.it.

Reperire la configurazione di una rete

Network API in VB

parte prima

Esistono diverse API che permettono di ottenere informazioni sulla nostra rete. E non solo…

Sappiamo tutti che i sistemi operativi Windows affidano la maggior parte dei compiti da svolgere a programmi che si appoggiano a funzioni "preconfezionate" contenute, a loro volta, in librerie dinamiche .DLL. E' quasi ovvio pensare dunque che, tutto quello che Windows consente di fare, dalla semplice gestione delle finestre, al recupero di informazioni e quant'altro viene messo a disposizione dell'utente/amministratore, sia possibile ottenerlo nella medesima maniera, ossia affidandosi alle centinaia di API fornite con il sistema operativo stesso. In realtà, un fondo di verità, relativamente a questa affermazione, esiste, poiché chi conosce questo "tipo" di programmazione, sa che le tantissime funzioni messe a disposizione da Windows consentono di effettuare operazioni spesso complicate (se non impossibili) da ottenere con i mezzi classici offerti dai più comuni linguaggi di programmazione. Le API di Windows, come già saprete, sono davvero moltissime, spesso piuttosto complicate da gestire, anche se consentono di ottenere risultati straordinari. Un esempio immediato è la "banale" funzione *SendMessage()*, attraverso la quale è possibile inviare ad una qualunque finestra di Windows un opportuno messaggio per notificarle il verificarsi di un particolare evento o per "ordinarle" di compiere qualche azione. Questa volta ci occuperemo di una particolare categoria di funzioni che si occupa di aiutare il programmatore nella risoluzione di problemi legati alle reti e, di conseguenza, ai protocolli che vi ruotano intorno (TCP/IP compreso, ovviamente).

LE IP HELPER API

Tra le tante librerie dinamiche rese disponibili da Windows, sicuramente qualcuno si sarà accorto della presenza della *IPHlpApi.dll*, una libreria dinamica il cui nome fa di certo subito sottointendere lo scopo. Attraverso le tantissime funzioni presenti all'interno di questa DLL, è possibile recuperare svariate informazioni ed effettuare diverse operazioni che, altrimenti, diventerebbero complicate da gestire, specie per linguaggi come Visual Basic.
Le varie tipologie di operazioni che possiamo compiere servendoci di questa libreria possono essere così suddivise:

- **Recupero delle informazioni relative alla configurazione di rete.**
- **Recupero di informazioni sul protocollo IP e ICMP.**
- **Configurazione degli adattatori di rete.**
- **Configurazione delle interfacce di rete e degli indirizzi IP.**
- **Utilizzo di ARP.**
- **Gestione delle notifiche di eventi riguardanti la rete.**
- **Recupero informazioni sul protocollo TCP ed UDP.**

Le funzioni appartenenti ad essa sono diverse decine e si appoggiano, per la maggior parte dei casi, a diverse strutture, spesso piuttosto complesse da utilizzare, ma che alla fine ci consentono di ottenere moltissime informazioni. Di seguito ecco solo una piccolissima parte delle funzioni che appartengono a questa libreria (solo le prime in ordine alfabetico) con una breve descrizione sul loro scopo:

AddIPAddress: consente di aggiungere un indirizzo IP.

CancelIpChangeNotify: elimina l'invio di notifiche sul cambiamento di indirizzo IP e d'instradamento

precedentemente richiesti attraverso *NotifyAddrChange* e *NotifyRouteChange.*

CreateIpForwardEntry: crea una route all'interno della route table.

CreateIpNetEntry: crea una nuova ARP entry.

IP_ADDR_STRING		
Next	Long	Puntatore alla successiva struttura IP_ADDR_STRING
IpAddress	String * 16	IP Address
IpMask	String * 16	Subnet Mask
Context	Long	Network table entry (NTE).

FIXED_INFO		
HostName	String * MAX_HOSTNAME_LEN	Nome HOST del PC
DomainName	String * MAX_DOMAIN_NAME_LEN	Nome del dominio di appartenenza
CurrentDnsServer	Long	DNS Server primario
DnsServerList	As IP_ADDR_STRING	Lista linkata dei server DNS
NodeType	Long	Node type del PC
ScopeId	String * MAX_SCOPE_ID_LEN	Netbios Scope ID
EnableRouting	Long	Routing abilitato/disabilitato
EnableProxy	Long	Proxy abilitato/disabilitato
EnableDns	Long	Risoluzione nomi con DNS

Tab. 1: Le strutture IP_ADDR_STRING e FIXED_INFO.

CreateProxyArpEntry: crea una nuova PARP entry relativamente ad un certo IP address.

DeleteIPAddress: cancella un indirizzo IP precedentemente creato con *AddIPAddress*.

DeleteIpForwardEntry: elimina una route entry all'interno della IP routing table.

DeleteIpNetEntry: elimina un'entry ARP.

DeleteProxyArpEntry: elimina una PARP entry.

DisableMediaSense: disabilita la funzionalità di media sensing di una determinata interfaccia di rete.

EnableRouter: abilita l'IP forwarding sul PC.

FlushIpNetTable: elimina tutte le ARP entry dall'ARP table.

GetAdapterIndex: ottiene l'indice di un adattatore di rete, partendo dal suo nome.

GetAdapterOrderMap: ottiene il valore di priorità per ciascuna interfaccia di rete.

Ovviamente è impensabile poter disquisire su tutte quante. Cercheremo invece di approfondire quelle che potrebbero risultare maggiormente utili, lasciando a chiunque fosse interessato, l'onere di approfondire l'argomento. A questo scopo sono molto utili i riferimenti a documentazione ed esempi reperibili direttamente dal sito della Microsoft o altrove su Internet. Prima di passare oltre, è bene rammentare che tutti gli esempi realizzati in questa serie di articoli sono stati testati su Windows 2000 Professional e che il linguaggio utilizzato per la costruzione dei progetti è stato Visual Basic 6.0 Professional.

☑ GLI STRUMENTI NECESSARI

Tutti gli esempi realizzati in questa serie di articoli sono stati testati su Windows 2000 Professional
Il linguaggio utilizzato per la costruzione dei progetti è stato Visual Basic 6.0 Professional.

LE FUNZIONI DEL PROGETTO

Questo primo progetto in Visual Basic è, per certi aspetti, molto semplice. Esso, infatti, consente di ottenere informazioni circa la configurazione della propria scheda di rete, sfruttando per questo soltanto due delle tante funzioni appartenenti a questa categoria. Tuttavia, quest'esempio consente di approfondire anche altri aspetti legati alla programmazione che potrebbero tornare utili anche per un altro genere di applicazioni. La prima funzione utilizzata all'interno del progetto è la *GetNetworkParams()* la cui sintassi, così come dichiarata all'interno di Visual Basic, è:

```
Public Declare Function GetNetworkParams Lib "IPHlpApi"
        (FixedInfo As Any, pOutBufLen As Long) As Long
```

dove:

FixedInfo: rappresenta un puntatore ad una struttura di tipo *FIXED_INFO* (mostrata in seguito) che non fa altro che raccogliere i dati relativi alla configurazione di rete del proprio PC. Il prefisso *FIXED*, davanti al nome della struttura, non è affatto casuale poiché, scopo di questa struttura, è proprio quello di memorizzare i dati di rete non strettamente legati ad un particolare network adapter installato sul PC. Alcune delle informazioni ritornate dalla funzione sono, ad esempio, il nome del dominio di appartenenza, il nome del PC, ecc.

pOutBufLen: questo valore *Long* specifica la dimensione in byte della struttura precedente.

Il valore di ritorno della funzione, in caso di successo, è zero. Contrariamente, assume un valore diverso e corrispondente ai possibili tipi di errore che potrebbero occorrere. La seconda funzione utilizzata, invece, è la *GetAdaptersInfo()* la cui sintassi, così come dichiarata all'interno di Visual Basic, è:

```
Public Declare Function GetAdaptersInfo Lib "IPHlpApi"
    (IpAdapterInfo As Any, pOutBufLen As Long) As Long
```

in cui:

IpAdapterInfo: questo parametro identifica un tipo di dato molto particolare. Esso, infatti, rappresenta un puntatore ad un buffer di tipo *IP_ADAPTER_INFO* (mostrato in seguito) che, attraverso un apposito elemento della struttura stessa, consente di "linkarsi" ad altre dello stesso tipo consentendoci di reperire le stesse informazioni per ogni apparato di rete installato. Naturalmente, nell'ipotesi classica più banale, ossia quella di un PC con una sola scheda di rete, questa struttura verrà sfruttata solo attraverso un'unica chiamata alla funzione.

pOutBufLen: analogamente a quanto visto per la funzione precedente, questo valore fa riferimento alla dimensione della struttura *IP_ADAPTER_INFO.*

Anche qui, analogamente a quanto visto per la funzione precedente, il valore di ritorno, in caso di successo, è zero. In caso contrario, invece, assume un valore diverso e corrispondente ai possibili tipi di errore che potrebbero occorrere.
Com'è possibile notare dai riquadri presenti, le strutture a cui fanno riferimento le due funzioni sono molto "particolari". La prima particolarità, come già accennato, riguarda il primo parametro della funzione *GetAdaptersInfo()* ossia una struttura di tipo *IP_ADAPTER_INFO.*

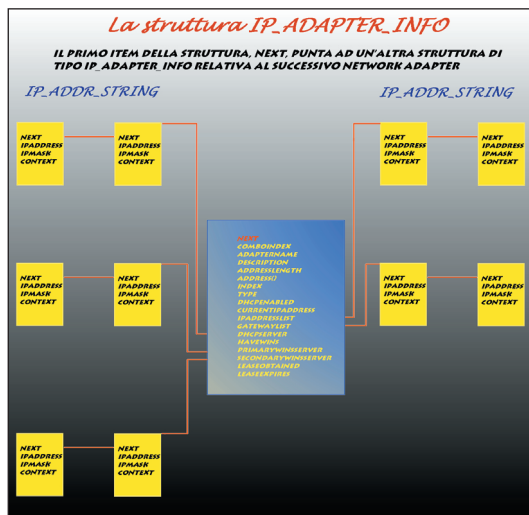



Fig. 1: La struttura IP_ADAPTER_INFO.

In prima "istanza" essa contiene i campi che memorizzano i dati relativi ai parametri di rete del primo "network adapter" più un campo identificato con l'etichetta *Next* che rappresenta il puntatore ad un'analoga struttura per il secondo apparato e così via. Questa lista linkata termina quando il valore *Next* è impostato a 0 nell'ultima struttura della catena. La seconda particolarità è quella che esistono degli elementi, all'interno delle due strutture principali, *FIXED_INFO* e *IP_ADAPTER_INFO*, che rappresentano, a loro volta, delle liste linkate di tipo *IP_ADDR_STRING*. Questa struttura, per alcuni aspetti molto complicata, consente, tuttavia, di ottenere informazioni circa ogni network adapter, ogni DNS, ogni gateway, ecc. senza che ci si debba preoccupare di quanti gateway sono configurati o di quanti IP Address, ecc. Oltre a queste due funzioni appartenenti alle API *IPHlpApi.dll,* il progetto VB ne sfrutta un'altra, estremamente importante ed allo stesso tempo, indispensabile per la gestione di queste informazioni: la funzione *CopyMemory().*
Essa appartiene alla libreria *Kernel32.dll* ed ha la seguente sintassi:

```
Public Declare Sub CopyMemory Lib "kernel32" Alias
        "RtlMoveMemory" (Destination As Any,
        Source As Any, ByVal Length As Long)
```

in cui:

Destination: indirizzo finale di memoria ove copiare il blocco dati.

Source: indirizzo iniziale di memoria facente riferimento al blocco dati da copiare.

Length: numero di byte da copiare.

Apparentemente una funzione siffatta può sembrare banale e, forse, inutile. Tuttavia è bene rammentare che le due funzioni viste precedentemente restituiscono "semplicemente" un riferimento di memoria ad una struttura che contiene i dati ricercati sottoforma di sequenza di byte che, in realtà, andrebbero letti secondo le specifiche viste prima.
A causa, quindi, della presenza di strutture linkate all'interno di altre, contenenti le nostre informazioni, è necessario riuscire a seguire questa catena di

Next	Long
ComboIndex	Long
AdapterName	String * MAX_ADAPTER_NAME_LENGTH
Description	String * MAX_ADAPTER_DESCRIPTION_LENGTH
AddressLength	Long
Address (MAX_ADAPTER_ADDRESS_LENGTH - 1)	Byte
Index	Long
Type	Long
DhcpEnabled	Long
CurrentIpAddress	Long
IpAddressList	IP_ADDR_STRING
GatewayList	IP_ADDR_STRING
DhcpServer	IP_ADDR_STRING
HaveWins	Boolean
PrimaryWinsServer	IP_ADDR_STRING
SecondaryWinsServer	IP_ADDR_STRING
LeaseObtained	Long
LeaseExpires	Long

Tab. 2: La struttura IP_ADAPTER_INFO

strutture, "formattando" di volta in volta tali successioni di byte. Tanto per renderci conto di quanto appena detto, diamo un'occhiata al seguente frammento di codice:

```
` Lista degli IP Address
txtNetworkInfo.Text = txtNetworkInfo.Text + vbCrLf +
  "IP Address 1): " & AdapterInfo.IpAddressList.IpAddress
txtNetworkInfo.Text = txtNetworkInfo.Text + vbCrLf +
   "Subnet Mask: " & AdapterInfo.IpAddressList.IpMask
` Altri IP Address?
pAdapter = AdapterInfo.IpAddressList.Next
NextItem = 2
Do While pAdapter <> 0
   CopyMemory pADDRStr, ByVal pAdapter,
                                          LenB(pADDRStr)
   txtNetworkInfo.Text = txtNetworkInfo.Text + vbCrLf
                    + "IP Address " & NextItem & "):
                                  " & pADDRStr.IpAddress
   txtNetworkInfo.Text = txtNetworkInfo.Text + vbCrLf
               + "Subnet Mask: " & pADDRStr.IpMask
   pAdapter = pADDRStr.Next
   NextItem = NextItem + 1
Loop
```

Esso si occupa di recuperare dalla struttura *AdapterInfo* (di tipo *IP_ADAPTER_INFO*), valorizzata precedentemente con una chiamata alla funzione *GetAdaptersInfo()*, la lista di tutti gli indirizzi IP configurati sulla propria scheda di rete.

L'elemento *IpAddressList* rappresenta inizialmente la prima struttura di una catena di strutture di tipo *IP_ADDR_STRING* e, di conseguenza, la lettura di *AdapterInfo.IpAddressList.IpAddress* e *AdapterInfo .IpAddressList.IpMask* non fanno altro che restituirci proprio il primo indirizzo IP e la relativa subnet mask configurati.

Naturalmente, dopo aver fatto ciò, quello che ci si deve chiedere è appunto se esistono altri elementi della catena o se, come accade più spesso, l'indirizzo IP configurato è uno solo.

Per ottenere quest'informazione, come già accennato all'inizio, ci si serve dell'item *Next* della struttura. Se questo elemento è pari a zero, ciò vuol dire che la catena è terminata, altrimenti occorre accedere alla struttura successiva. Il parametro *Next*, in questo caso, conterrà il riferimento al successivo elemento di questa catena.

Adesso entra in gioco il ciclo *DO… WHILE* che consente di effettuare, ricorsivamente, quest'operazione. La prima istruzione utilizzata è, appunto, quella che lancia la funzione *CopyMemory()*. Il suo scopo è quello di copiare una sequenza di byte pari alla lunghezza di una generica struttura *IP_ADDR_STRING*, partendo dall'indirizzo identificato dall'item *Next*, in un'analoga struttura di tipo *IP_ADDR_STRING*. Così facendo, il risultato ottenuto sarà stato proprio quello di "convertire" una sequenza di byte "apparentemente illeggibile" in una forma "riconosciuta e gestibile".

A questo punto, recuperati nuovamente i dati relativi all'indirizzo IP ed alla subnet mask voluta, il discorso di ripete ricorsivamente sino a quando il parametro *Next* non mostra come valore zero.

Next	Puntatore al successivo adapter
ComboIndex	Riservato
AdapterName	Nome dell'adattatore
Description	Descrizione dell'adattatore
AddressLength	Lunghezza del MAC Address dell'adattatore
Address	MAC Address
Index	Indice dell'adattatore rispetto alla lista degli adattatori
Type	MIB_IF_TYPE_OTHER = 1 MIB_IF_TYPE_ETHERNET = 6 MIB_IF_TYPE_TOKENRING = 9 MIB_IF_TYPE_FDDI = 15 MIB_IF_TYPE_PPP = 23 MIB_IF_TYPE_LOOPBACK = 24 MIB_IF_TYPE_SLIP = 28
DhcpEnabled	Indica se abilitato il DHCP
CurrentIpAddress	Riservato
IpAddressList	Lista degli indirizzi IP
GatewayList	IP Address del default gateway
DhcpServer	IP Address del DHCP
HaveWins	Valore booleano che indica o meno l'utilizzo di WINS
PrimaryWinsServer	WINS Server primario
SecondaryWinsServer	WINS Server secondario
LeaseObtained	Lease Time
LeaseExpires	Expire Lease Time

Tab. 3: La struttura IP_ADAPTER_INFO mantiene informazioni sugli adattatori di rete installati sul PC. Ecco una descrizione sommaria sul significato di ogni campo.

UNO SGUARDO ALL'ESEMPIO

Il progetto realizzato, come già accennato, è strutturato in maniera molto semplice. Esso è formato da una sola form principale e da un modulo per la dichiarazione delle variabili, delle funzioni e delle strutture utilizzate al suo interno.

All'avvio del form principale viene richiamata una funzione, denominata *GetNetworkInformation()*, che si preoccupa di reperire tutte le informazioni necessarie per poi mostrarle opportunamente a video.

La struttura di questa funzione è stata semplificata per consentire una rapida lettura ed un più agevole apprendimento del codice adottato. Sicuramente la strada migliore sarebbe stata quella di "spezzare" il codice in sottofunzioni aventi il compito, ad esempio, di leggere strutture di tipo *IP_ADDR_STRING* o gestire gli errori ritornati dalle funzioni, ma questo credo avrebbe reso sicuramente la lettura meno scorrevole.

Una volta richiamata la funzione *GetNetworkInformation()*, vengono inizializzate diverse variabili, tra

Fig. 2: *L'interfaccia principale del programma.*

le quali alcune strutture del tipo finora visto.
Un dettaglio importante e molto particolare circa l'implementazione adottata, è il modo nel quale strutture di tipo *IP_ADAPTER_INFO* e *FIXED_INFO* sono dimensionate e, successivamente valorizzate. Infatti, se riprendiamo per un attimo una delle due API, ad esempio la *GetNetworkParams()*, ci accorgeremo che la prima volta che essa è richiamata, il codice è del tipo:

```
' Ottieni I dati "FIXED" del PC. Scatena a questo proposito
' una chiamata "errata" che riporterà, come
' conseguenza, la corretta dimensione della struttura
  FixedInfoSize = 0
  Errore = GetNetworkParams(ByVal 0&, FixedInfoSize)

If Errore <> 0 Then
   ' Gestisci l'errore
   Select Case Errore
          'Case ERROR_BUFFER_OVERFLOW: '...
          'Case ERROR_INVALID_PARAMETER: '...
          'Case ERROR_NO_DATA: '...
          'Case ERROR_NOT_SUPPORTED: '...
   End Select
End If

' Ridimensiona la struttura che conterrà i dati in
' maniera corretta
  ReDim FixedInfoBuffer(FixedInfoSize - 1)

' Ricava i dati
  Errore = GetNetworkParams(FixedInfoBuffer(0),
                                         FixedInfoSize)
```

Contrariamente ad analoghe funzioni API, esse necessitano, come secondo parametro, della corretta dimensione della prima struttura, passata loro come primo parametro. A questo proposito, pertanto, si comportano in maniera molto particolare ossia, quando vengono lanciate con un parametro *pOutBufLen* contenente un valore insufficiente ad identificare la dimensione della prima struttura, esse ritornano un codice di errore identificato dalla costante simbolica *ERROR_BUFFER_OVERFLOW* e, conseguentemente, valorizzano con la *pOutBufLen* la dimensione corretta. Pertanto, la prima chiamata del tipo:

```
FixedInfoSize = 0
Errore = GetNetworkParams(ByVal 0&, FixedInfoSize)
```

permette proprio di ottenere questo risultato e, successivamente, ci consente di richiamare la funzione con i parametri corretti.
A parte questo "piccolo" dettaglio, il resto del programma credo sia piuttosto semplice da comprendere e non credo occorra dire nient'altro in proposito.

AddIPAddress	GetNumberOfInterfaces
CancelIpChangeNotify	GetPerAdapterInfo
CreateIpForwardEntry	GetRTTAndHopCount
CreateIpNetEntry	GetTcpStatistics
CreateProxyArpEntry	GetTcpStatisticsEx
DeleteIPAddress	GetTcpTable
DeleteIpForwardEntry	GetUdpStatistics
DeleteIpNetEntry	GetUdpStatisticsEx
DeleteProxyArpEntry	GetUdpTable
DisableMediaSense	GetUniDirectionalAdapterInfo
EnableRouter	IcmpCloseHandle
FlushIpNetTable	Icmp6CreateFile
GetAdapterIndex	IcmpCreateFile
GetAdapterOrderMap	IcmpParseReplies
GetAdapterAddresses	Icmp6ParseReplies
GetAdapterInfo	Icmp6SendEcho2
GetBestInterface	IcmpSendEcho
GetBestInterfaceEx	IcmpSendEcho2
GetBestRoute	IpReleaseAddress
GetFriendlyIfIndex	IpRenewAddress
GetIcmpStatistics	NhpAllocateAndGetInterfaceInfoFromStack
GetIcmpStatisticsEx	NotifyAddrChange
GetIfEntry	NotifyRouteChange
GetIfTable	RestoreMediaSense
GetInterfaceInfo	SendARP
GetIpAddrTable	SetIfEntry
GetIpErrorString	SetIpForwardEntry
GetIpForwardTable	SetIpNetEntry
GetIpNetTable	SetIpStatistics
GetIpStatistics	SetIpTTL
GetIpStatisticsEx	SetTcpEntry
GetNetworkParams	UnenableRouter

Tab. 4: Ecco le funzioni che fanno parte della libreria IPHlpApi.dll

CONCLUSIONI

Le due funzioni appena viste ci consentono di reperire informazioni circa la configurazione della nostra rete in maniera molto semplice ed efficace. La prossima volta avremo modo di analizzare altre funzioni "particolari" facenti parte di questa libreria, che risulteranno senza ombra di dubbio altrettanto utili ed interessanti.

Francesco Lippo

La programmazione dei SONY AIBO

Vedere il risultato dei nostri programmi realmente funzionante, può dare delle soddisfazioni. Ma vedere il nostro programma che si alza su quattro zampe, scodinzola e se ne va, che sensazioni può suscitare?

Probabilmente vi sarà capitato di vedere in TV, o su qualche rivista specializzata, le foto di un grazioso cagnolino dall'aspetto "meccanico". Un piccolo robot quadrupede che farebbe la gioia di qualsiasi bambino (e anche di chi bambino non lo è più da un pezzo). Stiamo parlando degli *AIBO (Artificial Intelligence BOt)* i mini-robot prodotti dalla Sony per scopi di intrattenimento, che ben presto si sono rivelati utilissimi nel campo della ricerca (ad es. in robotica) in quanto forniscono una piattaforma comune di sviluppo, ben supportata dal gigante industriale giapponese e a costi accessibili (relativamente ai costi necessari per la progettazione e costruzione di un robot "in proprio", ovviamente...). Nonostante il loro aspetto da giocattolo, infatti, questi robot sono un vero e proprio concentrato di tecnologia e possiedono al loro interno, oltre ai motori necessari per il movimento della testa, della coda e delle zampe, anche una serie di sensori per la "percezione" del mondo esterno, come ad esempio:

- **una videocamera a colori montata sulla punta del naso**
- **dei sensori di pressione sotto le zampe**
- **un sensore di distanza**
- **sensori interni di accelerazione, vibrazione e temperatura**
- **un microfono stereo.**

Inoltre possono comunicare all'esterno tramite particolari LED posti sulla testa (che simulano degli occhi), nonché tramite un piccolo speaker che permette loro di riprodurre dei suoni.
Una caratteristica molto interessante è costituita dalla possibilità di montare sugli AIBO delle schede di rete di tipo *wireless* (senza fili), per la comunicazione con dei normali PC. Già, vi starete chiedendo, ma perché dovrebbero comunicare con un PC? Beh, la risposta è semplice: il pezzo forte di questi aggeggi (almeno dal punto di vista di noi programmatori :-) è proprio il fatto che sono programmabili! é cioè possibile, tramite delle opportune *API (Application Programming Interface)*, scrivere del codice che permetta al robot di compiere determinate azioni, magari sfruttando come input i segnali prodotti dai sensori, cosa che lo rende a tutti gli effetti un *"agente autonomo"*, secondo la definizione cara a chi si occupa di Intelligenza Artificiale; il robot diventa cioè in grado di prendere delle "decisioni" (per quanto semplici) senza alcun intervento di tipo umano dall'esterno. La capacità elaborativa è assicurata da un microprocessore interno di tipo MIPS (la cui frequenza varia a seconda del modello), mentre la memoria di massa è costituita da una schedina di memoria rimovibile (di tipo MemoryStick, sempre prodotta da Sony) che

☑ AIBO
Nonostante il loro aspetto da giocattolo, questi robot sono un vero e proprio concentrato di tecnologia e possiedono al loro interno, oltre ai motori necessari per il movimento della testa, della coda e delle zampe, anche una serie di sensori per la "percezione" del mondo esterno.

***Fig. 1:** Il cane robot AIBO.*

è destinata a contenere il codice sviluppato e i dati necessari al funzionamento (ad es. file di testo, file sonori ecc.). Già ma come funziona tutto ciò? Come si usano queste API? Questo articolo è fatto apposta per spiegarvelo!

IL CANE E LE API

La programmabilità degli AIBO è assicurata da uno *SDK (Software Development Kit)* chiamato *OPEN-R*. OPEN-R è una piattaforma di sviluppo che costituisce un *layer* per la programmazione dei robot, un po' come, ad esempio, le librerie OpenGL costituiscono una piattaforma per la programmazione grafica sui comuni PC. OPEN-R si basa sul linguaggio C/C++ ed è liberamente scaricabile dal sito che trovate nel box a lato, previa registrazione gratuita.
Il fatto che si possa utilizzare un linguaggio di tipo *Object Oriented*, rende ovviamente più facile la vita allo sviluppatore, che può sfruttare tutti i vantaggi della programmazione a oggetti, primo fra tutti la *modularizzazione* del software. Il concetto di modularizzazione è tuttavia, in questo ambito, ancora più accentuato, in quanto, sviluppando con OPEN-R, non si ha il concetto di "*programma che utilizza degli oggetti", quanto piuttosto quello di "oggetti indipendenti che comunicano tra loro"*. Per visualizzare la situazione è utile pensare al funzionamento dei più moderni sistemi operativi per PC (UNIX /Linux, Windows ecc.), i quali permettono l'esecuzione contemporanea di più programmi; i programmi in esecuzione vengono chiamati *processi*. In questo modo è possibile ad esempio utilizzare un *word processor* mentre si ascolta musica in *mp3* e si scarica la posta elettronica, il tutto in contemporanea, col sistema operativo che gestisce chi, e per quanto tempo, deve utilizzare le risorse del computer (CPU, memoria, disco rigido ecc.). Negli AIBO la situazione è del tutto analoga e gli oggetti "vivono di vita propria" compiendo le loro azioni e comunicando tra loro. Possiamo pensare di schematizzare le caratteristiche salienti degli oggetti OPEN-R in questo modo:

Ogni oggetto funziona in maniera "concorrente" con altri oggetti - È quanto detto in precedenza: gli oggetti sono come dei "processi" che funzionano in contemporanea, sono tutti attivati all'accensione del robot e distrutti al suo spegnimento.

Gli oggetti comunicano tra loro - Il sistema di comunicazione è quello *"a scambio di messaggi"*: in pratica se l'oggetto *A* deve dire qualcosa all'oggetto *B*, gli manda un messaggio (cioè una struttura dati opportunamente configurata). Se l'oggetto *B* è pronto per leggere il messaggio, lo legge, altrimenti il messaggio viene messo in una *coda dei messaggi* e verrà letto in seguito, quando *B* finirà di fare quello che sta facendo. In questa situazione l'oggetto *A* è chiamato *Subject*, mentre l'oggetto *B* è detto *Observer*. Ogni messaggio contiene sostanzialmente due informazioni: una serie di dati e un numero intero detto *Selector*. Il *Selector* specifica quale è la funzione da attivare alla ricezione del messaggio, come spiegato nel successivo punto. Alla funzione attivata saranno passati come argomenti i dati del messaggio

Un oggetto ha dei "punti di accesso" multipli Questo potrebbe sembrare un po' ostico ma in realtà è una cosa molto semplice. I comuni programmi hanno (di solito) un unico punto di accesso, che è generalmente la funzione *main()*. Quando al sistema operativo viene detto di lanciare un programma, esso comincia a eseguirlo partendo dalla prima istruzione di questa particolare funzione. Possiamo vedere un oggetto OPEN-R come un programma che ha varie funzioni *main()* (ovviamente chiamate in maniera diversa!) che possono attivare l'oggetto. Ad es. supponiamo di avere un oggetto con le due funzioni *FaiPassoAvanti()* e *FaiPassoADestra()* e che entrambe siano dei "punti di accesso". Queste funzioni potrebbero essere richiamate da altri oggetti mediante messaggi nella coda dei messaggi (tramite il corrispondente *Selector*), in questo caso verrebbe eseguita per prima la funzione relativa al messaggio che si trova più avanti nella coda.

CORE CLASSES

Come di certo avrete capito, la definizione di oggetto OPEN-R è sicuramente differente dalla definizione di oggetto in un linguaggio di programmazione *Object Oriented*. Un oggetto OPEN-R è più vicino, come funzionamento, alla definizione di "programma vero e proprio" che non a quella di "tipo di variabile personalizzato". Ma è così per tutti gli oggetti presenti nel software che gira sugli AIBO? Tutti gli oggetti che definiamo, anche quelli che magari hanno pochi campi e sono usati solo per rappresentare una semplice struttura dati, devono avere la dignità di "programma"?
Evidentemente non è così. È infatti possibile definire tutte le classi che vogliamo e istanziare i relativi oggetti, senza che questi siano trattati come dei programmi autonomi; questo, ovviamente, consente una notevole flessibilità e un rispar-

☑ ROBOCUP

Uno degli utilizzi più interessanti degli AIBO è quello della ricerca scientifica in vari campi. Proprio a questo scopo, ogni anno, è organizzata una competizione internazionale di cani-calciatori, chiamata RoboCup. In questa competizione gli AIBO sono programmati per giocare a pallone su un campo di calcio in miniatura e si affrontano, quattro contro quattro, nelle squadre formate da varie università nel mondo.
www.robocup.org.

mio di risorse elaborative non indifferente.
Ma allora, come fare per distinguere gli oggetti OPEN-R dai normali oggetti C++? Gli oggetti OPEN-R sono particolari istanze di quelle che vengono chiamate *Core Classes (CC = classi nucleo)* e hanno una serie di particolarità che li rendono differenti dai normali oggetti C++. Innanzitutto è possibile istanziare solo un oggetto per ogni singola CC. Inoltre un oggetto di una CC deve:

essere derivato dalla classe OObject - Questa è una classe predefinita che implementa lo schema della logica di funzionamento delle CC;

implementare quattro particolari funzioni - Chiamate *DoInit(), DoStart(), DoStop()* e *DoDestroy();*

contenere i riferimenti agli oggetti Subject e Observer - Con i quali andrà a comunicare; (eventualmente) *definire delle funzioni "entry point" invocabili dalle altre CC.* Questo non è obbligatorio ma è evidente che una CC che non riceve e non trasmette messaggi non è di grande utilità.

Lo "scheletro" del codice di una CC è quindi, pressappoco, il seguente:

```
#include <OPENR/OObject.h>
#include <OPENR/OSubject.h>
#include <OPENR/OObserver.h>
#include "def.h"
class EsempioDiCC : public OObject
{
public:
  //Costruttore e distruttore
  EsempioDiCC();
  virtual ~EsempioDiCC() {}
  //Riferimenti agli oggetti Subject e Observer
  OSubject* subject[numOfSubject];
  OObserver* observer[numOfObserver];
  virtual OStatus DoInit(const OSystemEvent& event);
  virtual OStatus DoStart(const OSystemEvent& event);
  virtual OStatus DoStop(const OSystemEvent& event);
  virtual OStatus DoDestroy(const OSystemEvent& event);
  //Altre funzioni "entry point"
  //...
private:
   //Funzioni e campi accessibili solo da *questa* CC
   //...
};
```

Vediamo di analizzare questo codice. I primi tre file header importati, tramite la *#include*, sono quelli relativi alle definizioni di oggetto OPEN-R *(OObject.h)* e di *Subject* e *Observer*; sono indispensabili nella definizione di una CC, in quanto definiscono la logica di comunicazione e la struttura stessa di un oggetto di tipo CC. Il file *def.h* è un file standard, generato automaticamente durante il (complicato) processo di compilazione del software OPEN-R; esso contiene, tra le altre cose, le definizioni di:

numOfSubject
numOfObserver

cioè il numero di Subject e Observer con cui la nostra CC avrà a che fare. I riferimenti a questi oggetti esterni sono mantenuti nei due array dinamici:

subject[]
observer[]

e si potrà comunicare con tali oggetti invocando, ad esempio, una funzione in questo modo:

```
observer[event.ObsIndex()]->AssertReady();
```

in cui *event* è l'evento che ha determinato la chiamata (il parametro delle quattro funzioni speciali), *ObsIndex()* restituisce il suo indice all'interno del vettore dinamico e *AssertReady()* è una funzione che comunica la disponibilità di un oggetto a ricevere messaggi. Ma a cosa servono queste fantomatiche quattro funzioni speciali? Esse sono chiamate dal sistema operativo dell'AIBO in determinati momenti, e precisamente:

DoInit() - È chiamata per tutti gli oggetti CC non appena si accende il robot. Serve per contenere il codice di inizializzazione dell'oggetto.

DoStart() - È chiamata per tutti gli oggetti CC non appena sono finite tutte le chiamate a *DoInit()*. In questo modo è possibile mettere in questa funzione del codice, con la certezza che tutti gli altri oggetti siano stati già inizializzati.

DoStop() - È l'opposto di *DoStart()* e viene invocata, per ogni oggetto, quando si preme il tasto di spegnimento dell'AIBO.

DoDestroy() - È l'ultima funzione eseguita da ciascun oggetto CC. Essa viene invocata quando per tutti gli altri oggetti è stata invocata *DoStop()*. Può contenere del codice di finalizzazione, però non può comunicare con gli altri oggetti, in quanto non si è sicuri che questi siano ancora "vivi".

Una cosa interessante da notare è che, essendo

☑ DOVE SCARICARE OPEN-R

È possibile scaricare l'OPEN-R SDK direttamente da Internet, a partire dal sito

www.jp.aibo.com/openr/

Per accedere alla sezione download è necessario registrarsi, operazione del tutto gratuita. I file da scaricare sono:

OPEN_R_SDK-(numero versione).tar.gz
Il kit di sviluppo

OPEN_R_SDK-docE-(numero versione).tar.gz
La documentazione in Inglese

OPEN_R_SDK-sample-(numero versione).tar.gz
Alcuni file di esempio

Per potere compilare il software, è necessario avere installato il compilatore gcc. Mentre sotto UNIX/Linux questo dovrebbe esserci di default, sotto Windows, di solito, è necessario installare un suo porting. Sul sito è disponibile per il download il famoso CygWin.

Fig. 2: *I mini AIBO impegnati in una competizione.*

DoDestroy() l'ultima funzione invocata per ciascun oggetto, il distruttore C++ dell'oggetto CC stesso non verrà mai chiamato. Infatti, come si può vedere, il corpo funzione di tale distruttore è vuoto. Questo implicitamente vuol dire che ogni oggetto OPEN-R non ha né la possibilità di terminare un altro oggetto suo simile, né può "suicidarsi", interrompendo la sua esecuzione; in altre parole il ciclo di vita degli oggetti OPEN-R comincia con l'accensione del robot e finisce col suo spegnimento. Per quanto riguarda il codice contenuto in queste funzioni si può dire che, generalmente, esso non necessita di molta fantasia per essere implementato, in quanto sono disponibili delle *macro* che consentono di effettuare tutte le inizializzazioni più comuni, che molto spesso sono anche le uniche ad essere davvero necessarie al momento in cui vengono invocate le funzioni *DoXXX()*. Un tipico esempio di implementazione può essere il seguente:

```
Ostatus EsempioDiCC::DoStart(const OSystemEvent&
                                          event)
{
ENABLE_ALL_SUBJECT;
ASSERT_READY_TO_ALL_OBSERVER;
return oSUCCESS;
}
```

Questa funzione, come del resto le altre tre di questa categoria, prende come parametro un riferimento a un oggetto di tipo *OSystemEvent*, che rappresenta un "evento" nel ciclo di vita di un programma OPEN-R, un po' come la pressione di un pulsante su un'interfaccia grafica è un "evento" per il programma che la gestisce. Tale oggetto contiene delle informazioni utili, come ad esempio l'autore della chiamata alla funzione (in questo caso non è un altro oggetto, ma il sistema operativo stesso).

Una cosa che si può notare è che molto spesso, nel codice sviluppato per gli AIBO, si fa uso di parametri passati per riferimento (tramite l'operatore *"&"*) e in sola lettura (parola chiave *"const"*), questo ovviamente è dovuto a motivi legati all'efficienza di esecuzione del codice, su una piattaforma hardware così ridotta come quella degli AIBO: passare oggetti per riferimento evita di copiare l'oggetto stesso in un'altra area di memoria, risparmiando tempo e spazio preziosi.
Seguono nel corpo funzione due macro:

ENABLE_ALL_SUBJECT - Serve ad abilitare tutti i *Subject* che fanno riferimento a questo oggetto.

ASSERT_READY_TO_ALL_OBSERVER - Serve a dare il segnale di "sono pronto" *(ASSERT_READY)* a tutti gli *Observer* di questo oggetto OPEN-R. Gli oggetti nella fase di comunicazione possono anche emettere un altro segnale, di significato opposto: *DEASSERT_READY*, che sostanzialmente dice all'oggetto cui si sta parlando: *"ora sono impegnato"*

Infine, notiamo che viene restituito un valore di tipo *Ostatus* che, in questo caso, è *oSUCCESS* (cioè: "operazione compiuta con successo") ma che potrebbe essere anche un altro valore predefinito (ad es. *oFAIL* per una operazione non portata a termine in maniera corretta) oppure uno definito dal programmatore per particolari scopi.

CONCLUSIONI

In questo articolo, che è il primo di una serie di due, abbiamo visto la struttura logica del funzionamento di un software per robot, scritto mediante OPEN-R. Abbiamo visto come funzionano gli oggetti CC e quali sono i meccanismi che regolano la comunicazione e lo scambio di messaggi tra gli oggetti stessi. Questi passi sono fondamentali per potere capire il meccanismo utilizzato dalla piattaforma software, e possono risultare anche utili per chi si voglia cimentare nella programmazione di sistema *multi-process* o *multi-thread* con il proprio PC, per realizzare comuni applicazioni. Dopo questa dose di teoria, quindi, vedremo nella prossima puntata qualcosa di un po' più pratico e anche, se volte, divertente della programmazione degli AIBO.
Non mancate quindi, e se avete suggerimenti, consigli o critiche scrivete pure all'indirizzo *alfredo.marroccelli@ioprogrammo.it* per esprimere la vostra.
Alla prossima puntata dunque!

Alfredo Marroccelli

☑ ACQUISTARE AIBO

Il nome AIBO è l'acronimo di A(rtificial) I(ntelligence) (ro)BO(t) e in giapponese significa compagno. È possibile acquistare AIBO ad una "modica" cifra che si aggira dai 899,00 ai 2.199,00 Euro. L'acquisto può essere inoltrato on-line visitando il sito web:

http://shop.sonystyle-europe.com

ON LINE

MS OFFICE RESOURCE

Un sito, rivolto agli utilizzatori dell'ambiente Office di Microsoft.
Anteprime, tutorial, download, news, supporto tecnico e tanto altro, a portata di click.

http://office.microsoft.com/home/default.aspx

PROGRAMMING TUTORIALS

Hai da sempre cercato un repository di tutorial sui linguaggi di programmazione? Eccolo!
Un sito esclusivamente dedicato ai tutorial per la quasi totalità dei linguaggi di sviluppo.
http://www.programmingtutorials.com/

VISUAL BASIC INTERNET PROGRAMMING

Per gli amanti del linguaggio VB un portate dedicato allo sviluppo di applicazioni Internet.
Tutorial ed esempi pratici su come realizzare software in grado di interfacciarsi ad Internet.

http://www.vbip.com/

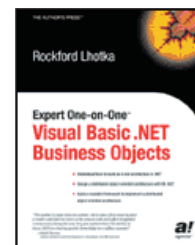
Biblioteca

JAVA - LA PROGRAMMAZIONE A OGGETTI

Java è ormai diventato un vero e proprio status symbol tra gli sviluppatori. I principali motivi per studiare un linguaggio di programmazione come Java sono il suo stretto legame con Internet, la multimedialità e la programmazione object oriented. Il libro si rivolge a tutti quegli utenti che si avvicinano per la prima volta ad un linguaggio a oggetti utilizzando un approccio semplice che guida all'apprendimento di concetti notoriamente tediosi: ereditarietà e polimorfismo in primis. Un'intera sezione viene dedicata allo sviluppo Web ed in particolare l'utilizzo degli applet Java. Si tratta sicuramente di un testo che non può passare inosservato a colui che vuole cimentarsi nella programmazione Java, volendo apprendere, da zero, tutti gli aspetti più salienti del linguaggio.

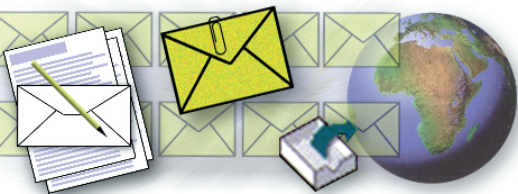
Difficoltà: Bassa • Autore: P. Gallo - F.Salerno • Editore: Mondadori Informatica
http://education.mondadori.it/libri **• ISBN: 88-8331-522-7 • Anno di pubblicazione: 2003**
Lingua: Italiano • Pagine: 288 • Prezzo: € 14,80

EXPERT ONE-ON-ONE VISUAL BASIC .NET BUSINESS OBJECTS

Rockford Lhotka, conosciuto per aver pubblicato numerosi testi a riguardo dello sviluppo di applicazioni distribuite, propone un nuovo testo, questa volta focalizzando l'attenzione sulla nuova piattaforma .NET. Il libro, suddiviso in tre diverse sezioni, analizza le architetture logiche e fisiche di un'applicazione, esplorando gli effetti di scalability, tolleranza di errori e prestazioni; l'attenzione viene focalizzata sugli strumenti di Visual Basic .NET per la progettazione di applicazioni distribuite; l'approccio è immediato, e grazie ad una serie di esempi pratici, il lettore sarà subito in grado di capire ogni aspetto inerente all'implementazione.

Difficoltà: Medio-Alta • Autore: R. Lhotka • Editore: Apress http://www.apress.com
ISBN: 1-59059-145-3 • Anno di pubblicazione: 2003 • Lingua: Italiano • Pagine: 725
Prezzo: $ 59.99

HTML & XML PASSO

Un testo base per chi si accinge ad utilizzare Html muovendo i primi passi nel web. L'approccio all'argomento è basilare e la lettura risulta scorrevole anche per chi non ha alcuna nozione sull'argomento.
Nei primi capitoli sono affrontati i temi principali: struttura e sintassi di una pagina Html, come inserire immagini, impostare lo stile del testo ed effettuare collegamenti ipertestuali. La seconda parte del testo si occupa di argomenti un po' più avanzati, infine due capitoli sono dedicati rispettivamente alle form ed al Dynamic Html.
Un ottimo manuale per chi non ha nozioni di Html e desidera iniziare a conoscere questo linguaggio per realizzare il proprio, semplice, sito Web per poi magari approfondire gli argomenti.

Difficoltà: Bassa • Autore: M.Morrison • Editore: Mondadori Informatica
http://education.mondadori.it/libri **• ISBN: 88-8331-485-9 • Anno di pubblicazione: 2003**
Lingua: Italiano • Pagine: 288 • Prezzo: € 24,80

INBox L'esperto risponde...

Estirpare il worm

Salve, prima di tutto vi faccio i complimenti per la rivista e poi passo subito al sodo... Nell'ultimo numero, quello di ottobre, ho letto con molto interesse l'articolo di Elia Florio "Fatti e misfatti di RPC". Purtroppo oltre l'interesse non sono riuscito ad andare, nel senso che non ho nessuna dimestichezza con gli argomenti trattati. Tuttavia il mio sistema (XP Home Edition) è afflitto, da qualche settimana, da un arresto di sistema iniziato da *NT/authority\\system* così come viene anche illustrato da voi. Ora mi chiedevo se posso fare qualcosa per risolvere questo problema che mi affligge la maggior parte delle volte che mi collego in rete.

Via e-mail

Risponde Elia Florio

Ciao, ti ringrazio per i complimenti e spero che la rivista continui sempre a soddisfare le aspettative di voi lettori. La rubrica degli "exploit" si rivolge in genere agli esperti di sicurezza, che conoscono le problematiche dei buffer overflow e dei bug scoperti ogni giorno nei software. Tuttavia, anche senza approfondire il codice, gli articoli di questa rubrica sono spesso interessanti da leggere, come nel tuo caso. Si tratta del worm "MSBlaster" (anche conosciuto come LovSan). In pratica quando ti connetti ad Internet, il tuo PC viene attaccato da qualche altro host infetto (connesso allo stesso provider) e riceve un pacchetto RPC malformato, in grado di causare il riavvio forzato. Per rimuovere il problema devi:

- aggiornare il tuo sistema WindowsXP con la patch *http://www.microsoft.com/downloads/details.aspx?displaylang=it&FamilyID=2354406C-C5B6-44AC-9532-3DE40F69C074* oppure, usando l'utility "windowsupdate.com" automatica
- rimuovere il worm dal tuo sistema usando l'utility *http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.removal.tool.html*
- proteggere la porta 135 del tuo PC con un personal firewall tipo ZoneAlarm o Norton Internet Security

Reperibilità dei componenti elettronici

Riguardo all'articolo (fantastico) di A. Margarese "Un telecomando per pilotare il PC", forse qualcuno ha avuto il mio stesso problema: non sono riuscito a trovare il ricevitore integrato di infrarossi siemens SFH-507-38. Così ho navigato qua e là in internet finché sono capitato sul sito *http://www.linuxguru.be*, dove "unidog" (che ringrazio) ha risposto ad un mio post sul forum che il sito ospita, dicendomi di dare un'occhiata a *http://www.lirc.org/receivers.html.* In questa pagina c'è una lista di ricevitori equivalenti, i componenti e lo schema per costruire il circuito di interfaccia con la porta parallela. Colgo l'occasione per fare i complimenti all'autore e alla rivista in generale, i cui articoli mettono alla portata di tutti argomenti che credevo per soli "eletti".

Matteo, via e-mail

Risponde Amedeo Margarese

Salve Matteo, purtroppo questo dipende solo ed esclusivamente dai negozi di elettronica ai quali ci si rivolge ed ahimé, soprattutto nelle città più piccole può non essere facile trovare alcuni componenti più ricercati. Nell'articolo viene indicata la possibilità di utilizzare un qualsiasi ricevitore integrato di infrarossi che funzioni intorno ai 36-38khz, rimandando però il lettore alla ricerca di suddette liste di equivalenza. Su molti siti, è possibile acquistare questi componenti on-line, anche se ordini di piccola entità possono rappresentare un problema.
Sul sito indicato dal forum si parla di un ricevitore infrarossi per porta seriale, con qualche differenza, quindi, da un ricevitore per porta parallela sia a livello circuitale che software. Dal punto di vista del software di comando, il problema viene trattato in maniera differente sia perché si parla di un altro sistema operativo sia perché questo argomento, pur essendo trattato da diversi siti in rete, viene raramente affrontato tramite l'uso degli interrupt come avviene nell'articolo da me redatto.

Librerie TAPI e ISDN

Salve, mi chiamo Claudio, leggo la vostra rivista con interesse e la trovo molto utile ed interessante, anche per uno poco esperto di programmazione come me. Vengo al dunque, è un po' che mi diletto con VB e quando vedo sulla rivista alcune proposte mi cimento a farle da me, per cercare di acquisire una maggiore dimestichezza con il linguaggio. Sulla rivista del mese scorso (n° 70), mi sono cimentato nella realizzazione del progetto "Controlla il tuo PC dal Telefonino. Purtroppo, il programma non mi funziona a dovere in quanto il modem che ho io è di tipo ISDN, per cui le informazioni contenute nel file di Registro "Dim ModemIDFromRegistry As Long..." non rilevano il mio modem (Tipo Hamlet ISDN). Vorrei sapere se cortesemente potreste darmi una dritta al riguardo.

Claudio, via e-mail

Risponde Elia Florio

Purtroppo le librerie TAPI utilizzate per realizzare il progetto (ModemTools della NetPoint) non funzionano correttamente con i modem ISDN, dispositivi che sono chiamati "modem" in maniera impropria (in realtà sono dei TA, ossia dei "Terminal Adapter"). L'ISDN-TA è un apparecchio che lavora su linee digitali, mentre il modem tradizionale è un convertitore di segnale che lavora prettamente su linee analogiche. Di conseguenza le librerie TAPI di Windows, per queste due categorie di apparecchi, sono leggermente diverse e spesso danno luogo a tali incompatibilità. Ti consiglio di testare il progetto con un qualsiasi altro modem analogico 56K; nelle nostre prove tutto ha funzionato per il meglio: la ricezione dei toni DTMF avviene correttamente e viene gestita dall'applicativo VB di esempio.
Cordiali saluti.

Intercettare un tasto in una procedura

Salve a tutti, Vivo a Trieste e sono un programmatore di siti Internet/intranet e di applicazioni in Delphi 4.
Ho un applicazione in cui in una form lancio una stampa particolarmente complessa (lancio una sere di elaborazioni nella procedura Button1Click. Avrei bisogno di intercettare un tasto (esempio F3 o ESC) dopo aver premuto il bottone per poter interrompere l'elaborazione.
Ho provato così...

```
procedure TForm1.Button1KeyUp(
        Sender: TObject; var Key: Word;
                    Shift: TShiftState);
begin
if key = vk_f3 then
begin
showmessage('Elaborazione interrotta
                        dall'utente! ');
      close;
   end;
end;
```

Ma non funziona... lo intercetta solo alla fine della procedura Button1Click.
Avete qualche idea?
Grazie

Raul • corwin56

La risposta di Salvatore Meschini

Utilizzando *Application.ProcessMessages* rendi possibile l'elaborazione della coda dei messaggi (e' quello che ti serve per non "bloccare" l'applicazione)... Per operazioni lunghe conviene inserire una chiamata a tale metodo per "restituire il controllo" all'utente.
Anche se secondo me la soluzione migliore resta la creazione di un nuovo thread per la stampa (come fa per esempio Microsoft Word)!
Il metodo ProcessMessages e' descritto nella documentazione di Delphi.
Esempio:

```
procedure TForm1.Button1Click(Sender:
                                TObject);
begin
Etichetta.Caption:='Inizio stampa";
Application.ProcessMessages;
                    // "Quasi-Multitasking"
sleep(10000); // Operazione lunga: STAM-
PA
Etichetta.Caption:='Fine stampa";
end;
```

Domandone da veri esperti

Ho iniziato la progettazione di un client di posta elettronica e ho ottenuto una prima bozza.
Vorrei fare in modo che la mia applicazione avesse la possibilità di iconificarsi nel senso che essa, quando viene iconificata, scompare lasciando traccia solamente con una piccola iconcina posta in basso a destra (dove ci sono le icone attive: Volume, Java (per chi ce l'ha)...
Praticamente l'applicazione dovrebbe sempre essere attiva ma non visibile e per farla riapparire dovrei cliccare sulla iconcina definita sopra. Non so se ho esposto bene la specifica
Purtroppo non ho idea di come si faccia!!!
Mi sapreste aiutare?
Per informazione, l' applicazione estende un JFrame.
Grazie per l'aiuto.

carMAN

La risposta di kyakan

Usare un tray icon in java non è molto facile, questo è dovuto al fatto che non sempre una applicazione java si trova su sistemi windows o comunque non è dette che ci sia sempre il tray!
Per integrare questa funzionalità bisogna ricorrere a programmi scritti in C e quindi interfacciarsi mediante una classe che dichiara alcune funzioni native che fanno riferimento alla libreria C.
Sul Web esistono diverse soluzioni già implmentate che permettono di gestire il tray icon su sistemi windows, cercando su google io ho trovato queste:

Tray4j (*http://www.volny.cz/mates1234/Tray4J.zip*)

trayicon presente in versione A (*http://jeans.studentenweb.org/java/trayicon/trayicon-1.7.6a.zip*)
e versione C (*http://jeans.studentenweb.org/java/trayicon/trayicon-1.7.5c.zip*)

Personalmente ho usato *Tray4J* ed è relativamente facile e completo allo stesso tempo, l'unico problema che ho riscontrato è che nel codice (è presente anche il sorgente java) c'è un System.out praticamente inutile (credo sia un debug rimasto).
Il problema lo puoi risolvere ricompilando il codice.
Naturalmente c'è da inserire una dll nella cartella bin del jdk usato, vicino all'interprete java.
Ciao!

IP address

È possibile conoscere l'IP address da VB (cioè da programma) dopo aver effettuato una connessione ad accesso remoto?
Grazie

pa_pier

La risposta di JBhemot

Certo ;-) un modo può essere utilizzando il componente Microsoft 'Winsock control'.
Quest'ultimo possiede una proprietà di nome 'LocalIP' che ti restituisce appunto il tuo indirizzo IP.
Saluti.

Client di posta in Java

Salve, vorrei sapere da voi quali api utilizzare per la gestione della posta e se mi potreste fornire qualche link utile sull'argomento, suggerire un tutorial...
Help!

JBhemot

La risposta di Carlo Pelliccia

Le API che fanno al caso tuo sono *http://java.sun.com/products/javamail/*. È tempo che mi ripropongo

anche io di scrivere un client e-mail in Java.
Prima o poi lo farò, e pensavo anche di proporre un tutorial per ioProgrammo sull'argomento.
Casomai confronteremo i nostri risultati, ok? :-)
Ciao!

Utilizzare i socket su rete con proxy

Vorrei sapere com si puo' aprire e utilizzare una connessione via socket quando devo mettere in comunicazione due PC appartenenti a reti divise da un Server Proxy.
Grazie

(Enrico

La risposta di Car1otta

Mmm ... o ti fai aprire la porta sul proxy dall'amministratore, o non c'e' possibilita' di riuscirci.
E' lo stesso problema che si ha quando da una Intranet si vuole utilizzare una chat ...
Il problema puo essere aggirato se su una delle due makkine c'e' un web server (e tu ci puoi mettere mani) ed il proxy ha una porta http aperta (di solito e' la 80) allora puoi mascherare la comunicazione come un flusso http ...

Come visualizzare un'immagine

Salve a tutti, ho un Db in Access con una tabella che contiene un campo in cui bisogna inserire un'immagine.
Quest'img dovrà essere visualizzata insieme agli altri campi in una pagina ASP.
Come posso fare?
E soprattutto è meglio che l'immagine del DB sia di tipo OLE o che sia solo testo?
Grazie a tutti

bianciuccio

La risposta di danyw3b

Evita l'immagine di tipo OLE nel db poichè le prestazioni calano notevolmente.
Anche io concordo sul fatto di salvare nel db solo l'url dell'immagine e quando vuoi visualizzarla fai inserire il percorso dell'immagine all'interno del tag img in questo modo:

```
<img src="<%=objRS(
        "CampoConPercorsoImmagine")%>">
```

...ma come si stampa???

Salve, ho questo seccantissimo problema...che non so proprio come risolvere.
Dovrei stampare il contenuto di un datatable ma non ho la più pallida idea su come fare...
Mi potreste aiutare??
Grazie 1000

albadur

La risposta di alip

Usa DataEnviroment come data source e DataReport per preparare un Modello di report (puoi usare anche il Drag and Drop) da *dataEnviroment* a *dataReport....* ovviamente devi prima preparare l'oggetto *Command* nel *DataEnviroment...*

Luminosità di un'immagine

Salve a tutti. Ho un piccolo problemino... dovrei salvare un'immagine con una luminosità diversa da quella originale, ma non riesco a trovare niente che possa aiutarmi...
Se qualcuno può darmi una mano, ne sarei molto felice.
Ciao e grazie a tutti...

corkor

La risposta di Juppiter

Ciao corkor, il problema che poni ha una soluzione semplice ma di difficile programmazione.
Fondamentalmente dovresti copiare nella memoria l'immagine con la chiamata API BitBlt, Quindi devi valutare pixel per pixel il filtro da applicare.
Se ogni pixel ha un valore RGB, è necessario valutare la trasformazione da imporre.
Ad esempio:

pixel(x1,y1) --> R(200),G(143),B(120)

Vuoi valutare ogni singolo componente del colore o tutti e tre insieme? Se uno di questi raggiunge il massimo (255) gli altri vengono bloccati? Una volta deciso questo puoi applicare una semplice trasformazione alla matrice che rappresenta l'immagine.
Alla fine puoi copiare l'immagine modificata nello stesso controllo o in un altro, oppure salvarla.
Detto così sembra semplice, ma non lo è affatto (perlomeno dal mio punto di vista), poiché Visual Basic ha delle serie limitazioni nella gestione della memoria.
Non è impossibile però.
Ti rimando ad un articolo esempio al seguente link:
http://www.vbaccelerator.com/codelib/gfx/imgproc.htm
CHEERS

Juppiter

Output DTS

Ho creato una DTS che mi fa delle select all'interno di un database. Come faccio a farmi ritornare, all'esecuzione della query una griglia o un file con i risultati?

jaco1982

La risposta di mchim

Nella connessione di destinazione potresti inserire come DATA SOURCE Microsoft excel 2000.
In questo caso il risultato della query viene inserito in excel in forma tabellare.

PER CONTATTARCI

e-mail: iopinbox@edmaster.it
Posta: Edizioni Master,
Via Cesare Correnti, 1 - 20123 Milano